NPS CS-92-001

NAVAL POSTGRADUATE SCHOOL
Monterey, California

TECHNOLOGY SURVEY AND PRELIMINARY DESIGN FOR SMALL AUV NAVIGATION SYSTEM

R.B. McGhee, J.R. Clynch, S.H. Kwak, and J.B.McKeon

March 1992

Prepared for:

NOSC Hawaii Laboratory
Kailua, HI 96734-0997

NAVAL POSTGRADUATE SCHOOL

Monterey, California

REAR ADMIRAL R. W. WEST, JR.
Superintendent

HARRISON SHULL
Provost

This report was prepared for the Naval Ocean Systems Laboratory, Kailua, HI and funded by the Naval Postgraduate School.

This report was prepared by:

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) NPSCS-92-001	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Computer Science Dept. Naval Postgraduate School 6b. OFFICE SYMBOL (if applicable) CS	7a. NAME OF MONITORING ORGANIZATION Dr. Randy Brill, NOSC Hawaii Laboratory
6c. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943	7b. ADDRESS (City, State, and ZIP Code) P.O. Box 997 Kailua, HI 96734-0997
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Naval Postgraduate School 8b. OFFICE SYMBOL (if applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER OM&N Direct Funding
8c. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
TECHNOLOGY SURVEY AND PRELIMINARY DESIGN FOR A SMALL AUV NAVIGATION SYSTEM

12. PERSONAL AUTHOR(S)
R.B. McGhee, J.R. Clynch, S.H. Kwak, and J.B.McKeon

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final	FROM 3/1/91 TO 9/30/91	March 1992	59

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	GPS, INS, IMV, Navigation, AUV missions Computer System Hardware, Real-Time Software

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Navigation of an Autonomous Underwater Vehicle (AUV) is a problem that has not been adequately solved. Although the inclusion of the Global Positioning System (GPS) into AUV navigation has been briefly examined before, this possibility is explored further in this report. GPS and Inertial Measurement System (INS) based navigation package offers many advantages for AUV navigation especially for transit and precise object location in shallow water.

This report provides background information on GPS and INS as they pertain to small AUV employment. Other required components are also examined as they pertain to small AUV employment. The use of the GPS/INS navigation package for AUV transits and precise object location work is presented. Two preliminary Small AUV Navigation System (SANS) designs with specified components are developed.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED	
22a. NAME OF RESPONSIBLE INDIVIDUAL Robert B. McGhee	22b. TELEPHONE (Include Area Code) (408) 646-2449	22c. OFFICE SYMBOL CS/Mz

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete

TECHNOLOGY SURVEY AND PRELIMINARY DESIGN FOR A SMALL AUV NAVIGATION SYSTEM

R. B. McGhee, J. R. Clynch, S. H. Kwak, and J. B. McKeon

Naval Postgraduate School
Monterey, CA 93923

1. INTRODUCTION

This report presents the findings of the first phase of an investigation [1] whose purpose was to determine the feasibility of developing a self-contained navigation and mission control package suitable for external attachment to a small autonomous underwater vehicle (AUV). For the purposes of this study, it was assumed that the vehicle is designed to accomplish a bottom survey in a hostile environment and that the mission therefore consists of three phases: a long range *transit* phase from a mother ship to the survey site, a short range *mapping* phase requiring precise navigation to obtain an appropriate coverage of the survey area, and a second long range *return* phase back to the mother ship. It was also assumed that the general shape and size of the vehicle corresponds roughly to an enhanced version of the NPS Model II AUV [2, 3, 4, 5]. The manueuvering capabilities anticipated for this vehicle are as follows: maximum speed = 10 kts, average speed = 3 kts, maximum pitch/angle = $\pm$ 60 degrees, angular rotation rate < 180 deg/sec(all axes), roll rotation angle unbounded.

In order to accomplish the transit and return phases of the mission, it is further assumed that the vehicle navigation accuracy requirements can be met by periodic GPS fixes using C/A (coarse acquisition) accuracy (50 meters rms). Between such fixes, vehicle navigation is accomplished by means of dead reckoning using instruments and a small computer contained in the mission package. At the survey site, the accuracy of the navigation system will be enhanced by recording of precise depth measurements along with angular rate and linear acceleration data obtained from a small solid-state inertial measurement unit (IMU). More frequent GPS fixes may also be used to further improve the accuracy of position estimation during the mapping phase of the mission.

An important aspect of the proposed system is the use of navigation data post processing to dramatically improve accuracy through the use of *differential* GPS [6]. This accuracy is preserved during submerged operations by an inertial navigation system (INS) based on *optimal position estimation* using an appropriate smoothing filter [7]. The use of post

processing differential GPS depends upon recording of GPS position or raw satellite data by the AUV mission package while in mapping mode. Upon completion of the return phase of a mission, this data can then be compared to similar data recorded at the mother ship to enable post processing determination of differential location with an accuracy of the order of 5 meters rms or better [6]. If the mother ship possesses military level GPS capability, then absolute positioning accuracy at high precision is thereby transferred to the mapping area as soon as processing of mission data is completed.

This investigation was guided by the following set of general design objectives:

a. Low Power Consumption. Operation from a small external battery pack for 24 hours is desirable.

b. GPS antenna exposure time in survey area should be minimized. Up to 30 seconds exposure allowed, but intervals between such exposures should be as long as possible, exceeding several minutes at a minimum.

c. The GPS antenna should present a very small cross section when exposed, and should not extend more than a few inches above the ocean surface.

d. For the mapping phase of a mission, system positioning accuracy of 10 meters rms or better is required with post-processing, submerged as well as surfaced.

e. During transit and return mission phases, C/A GPS data provides sufficient navigation accuracy when surfaced. When submerged, magnetic heading data provides sufficient accuracy for course control to next GPS fix.

f. Total volume not to exceed 120 inches. Elongated, streamlined packaging preferred.

The remainder of this report first addresses the state of the art regarding IMU and INS systems, GPS receivers, and computer hardware and software. This is followed by the presentation of two alternative system designs. Finally, the report concludes with a summary and recommendations for further work relative to the proposed small AUV navigation system (SANS). The overall conclusion is that the specifications above can be met at a hardware cost not exceeding $50K in FY 94. An interim system meeting

essentially all performance requirements except for the ability to accomplish continuous bottom mapping could be constructed in FY 92 at a hardware cost not exceeding $25K. This interim system achieves the necessary accuracy by means of a "pop-up" maneuver to the surface to obtain a GPS fix whenever an object of interest is found on the bottom.

2. INERTIAL NAVIGATION

2. 1 Overview

Inertial navigation systems (INS) permit the determination of position by means of double integration of sensed acceleration. The first such integration can be achieved using measured acceleration expressed either in body coordinates or in earth coordinates, leading to velocity components in the corresponding coordinate system. However, in either case, the second integration must be accomplished in earth coordinates, since it is generally only the position of a vehicle with respect to the earth that is significant to mission execution.

Early inertial navigation systems typically made use of *stable platforms* carrying three single-axis accelerometers aligned to measure acceleration directly in earth-fixed coordinates, thereby eliminating the need for coordinate conversion prior to integration. High precision navigation systems are still generally of this type, but the cost, weight, and volume associated with the gimbal mechanisms and servo drives needed to maintain platform orientation relative to the earth during vehicle motion have provided a strong motivation to searches for alternative approaches. The primary alternative today is the *strap down* class of systems in which an inertial measurement unit (IMU) containing three angular rate sensors and three linear accelerometers is mounted rigidly to the vehicle body. The current state of the art in such systems is typified by the Honeywell product documented in [8]. Due to recent order of magnitude decreases in cost in strapdown systems accompanied by a concurrent order of magnitude of increase in accuracy, only this type of system will be considered further in this chapter of this report.

2.2 Angular Rate Sensing

2.2.1 Background

There are presently at least five competitive technologies being pursued for angular rate sensing for strapdown IMUs. These are: rotating ("iron") gyros, fiber optic gyros, ring laser gyros, vibratory rate sensors, and fluidic rate sensors. Only the first of these technologies makes use of a rotating mass as an orientation reference. This approach is by far the oldest, having been understood for more than a century. It is based on the principle that the precession rate of a spinning rigid body or "gyroscope" is proportional to applied torque. That is

$$\frac{d}{dt} I \vec{\omega} = \vec{\tau} \qquad (2\text{-}1)$$

where I is the moment of inertia matrix ("inertia tensor") of a solid body, $\vec{\omega}$ is its angular rate vector,and $\vec{\tau}$ is a vector of applied torques. By keeping the magnitude of $\vec{\omega}$ constant while forcing its orientation to remain aligned with a specified axis attached to the body of a vehicle, the required $\vec{\tau}$ provides an estimate of the component of vehicle angular rotation rate orthogonal to $\vec{\omega}$. Typically, three single-axis rate gyros are used in IMU's, although two axis gyros are also found in some systems.

While many strapdown systems currently in use utilize rotating rate gyros, such devices inherently involve high cost, high power requirements, short lifetimes, and poor accuracy. A pertinent example is the three-axis rate gyro package currently employed by the NPS Model II AUV. This component is roughly the size of the entire navigation package proposed in this study, requires more power than will be available, possesses an unacceptable drift rate for inertial navigation, and has a lifetime of only about 200 hours [3]. The cost of this sensor was almost $10K when purchased three years ago.

Although improvements in rotating rate gyros continue, this is essentially a mature technology and the authors of this report have concluded that competing approaches to rate measurement offer more promise for near term realization of the small AUV navigation package design goals. Consequently, in what follows, no further consideration is given to rotating rate sensors.

2. 2. 2 Fiber Optic Gyros and Ring Laser Gyros

Both fiber optic gyros and ring laser gyros measure rotational rate about a single axis by making use of changes in the apparent speed of light induced by rotation of a dielectric medium. The difference in the two technologies relates mainly to the type of medium used to confine the path of a laser beam to a closed circuit. In the case of fiber optic gyros, a coil of optical fiber is used to obtain a long path length, while ring laser gyros utilize a much shorter path in a rigid triangular or quadrilateral shaped solid glass body with precision mirrors machined at each corner. The detection of changes in the travel time of laser light due to angular rotation is based on interferometry for either type of system. However, the details are complicated and are irrelevant to the purpose of this study. Rather, in what

follows, we will be concerned only with the parameters of such systems pertinent to small AUV navigation.

Until very recently, fiber optic gyros (FOG) were thought to be the only reasonable inexpensive alternative to rotating rate gyros. This was due to the compactness and low cost of the fiber coil and the relative simplicity of the associated electronics in comparison to that required for ring laser gyros. A typical example of a low cost IMU based on FOG technology is the Litton LN-200, which provides a complete missile IMU in a package with dimensions 5 x 5 x 3 inches, weighing 3 pounds, consuming 15 watts, and exhibiting 20,000 hrs MTBF. Pilot production of this unit is anticipated in approximately one year with full production costs eventually in the $10K to $20K range [9]. Table 2-1 gives advertised performance for this unit. It should be noted from this table that this IMU provides one integration, so that incremental velocity and incremental rotation are available as outputs. A very similar system with comparable size, performance, and price is under development by Honeywell for torpedo applications [10].

Table 2-1: Litton LN-200 IMU Performance

	Gyro	**Accelerometer**
Range	1000 deg/sec	40g
Bias	1 deg/hr	200 μg
Scale Factor	100 ppm	300 ppm
Output	$\Delta\theta$	ΔV
Noise	0.03 deg/$\sqrt{\text{hr}}$	

Somewhat surprisingly, very recent developments in ring laser gyro technology have lead to a prediction that such systems will be competitive in all respects with fiber optic gyros within about one year. Specifically, the HG-1700 IMU, being developed by Honeywell under a tri-service contract, uses triangular ring laser gyro sensors with a leg length of only approximately 1.5 inches to obtain a pancake shaped IMU 5 1/4 inches in diameter and 2 1/4 inches thick. Volume production costs of a complete IMU are expected to be on the order of $10K, with performance characteristics equal to or superior to those listed in Table 2-1 [11].

2.2.3 Vibratory and Fluidic Rate Sensors

Vibratory rate sensors make use of a tuning fork which resists rotation due to the linear momentum of its arms. Specifically, a *Coriolis* force arises in the base of the fork as it is rotated, and the magnitude of the resulting torque is proportional to angular rate. The current state of the art in this type of rate sensor is represented by the Systron Donner QRS sensor designed as a replacement for rotating rate gyros for aircraft and missile autopilots and for short-term inertial guidance. Each QRS sensor occupies less than 1 cu.in and requires less than 0.8 watts power. Cost is estimated to be in the range of $1K per axis or less by the end of 1992 [12]. Volume production of this sensor for the Maverick missile began in the 4th quarter of 1991. In many respects this sensor is ideal for the small AUV navigation system except that its drift rate is very sensitive to temperature, with a specification of 0.01 deg/sec/deg C. As subsequent error analysis will show, this drift rate appears to rule out this type of gyro for the SANS system, although further research is needed to establish this judgment definitively. Such work is currently underway at NPS [7].

Fluidic rate sensors also make use of Coriolis forces by measuring the deflection of a stream of helium gas caused by angular rotation of a sensor nozzle. While such systems are physically small and rugged, and are commercially available at a cost of $2K per axis or less [13], their output linearity is in the range of one percent of full scale. This makes this class of devices unsuitable for strapdown inertial navigation applications, at least at their current state of development.

2.3 Translational Acceleration and Velocity Sensing

2.3.1 Acceleration Sensing

All Accelerometers make use of some type of proof mass to sense acceleration according to Newton's Second Law,

$$\vec{f} = m (\ddot{\vec{x}} - \vec{g}) \qquad (2\text{-}2)$$

where m is the mass of a rigid body, $\ddot{\vec{x}}$ is translational acceleration, $\vec{g}$ is gravitation acceleration, and $\vec{f}$ is the *specific force* acting on the proof mass. In all accelerometers, the

proof mass is constrained to a fixed position relative to a housing, $\vec{f}$ is measured by some means, and Eq (2-2) is used to obtain

$$\ddot{\vec{x}} = \frac{\vec{f}}{m} + \vec{g} \qquad (2\text{-}3)$$

$$= \ddot{\vec{x}}_a + \vec{g} \qquad (2\text{-}4)$$

where $\ddot{\vec{x}}_a$ is linear acceleration as indicated by the accelerometer. This equation reveals a central difficulty in inertial navigation; namely, accelerometers do not sense gravitational acceleration. That is, imagine an accelerometer located at a fixed location on Earth. Then (for the moment ignoring earth rotational effects), $\ddot{\vec{x}} = 0$, so from Eq. (2-4)

$$\ddot{\vec{x}}_a = -\vec{g} \qquad (2\text{-}5)$$

That is, the indicated acceleration is due entirely to the specific force acting on the proof mass. Putting it another way, an accelerometer in a free fall or "weightless" condition produces a zero reading whereas its actual acceleration is exactly that due to gravitation attraction by the Earth; i.e., the true acceleration is "one g".

The above discussion shows that accelerometers for inertial navigation must be very linear with a known scale factor and a small offset. As Table 2-1 shows, IMU grade instruments satisfy these requirements with an accuracy of a few parts in ten thousand. This accuracy is achieved by relatively inexpensive silicon accelerometers [8, 9], typically costing only a thousand dollars or so per axis.

While an accuracy of one part in ten thousand sounds very precise, in fact, since the magnitude of gravitational acceleration is 32.2 ft./sec^2, this corresponds to an acceleration error of.0032 ft./sec^2 or 100 μg. If this error takes the form of a constant bias (an oversimplification, but still useful for error bounding), then, the time required for a 5 meter drift in position estimation is easily computed from

$$1/2\ at^2 = \frac{0.0032}{2} t^2 = 16 \text{ ft.} \qquad (2\text{-}6)$$

or

$$t = \left(\frac{32}{0.0032}\right)^{1/2} = 100 \text{ sec} \qquad (2\text{-}7)$$

This result clearly indicates the need for careful bias estimation and periodic GPS fixes to adequately determine small AUV position with respect to the specifications adopted for this study.

2.3.2 Velocity Sensing

While the above analysis provides a measure of the effect of accelerometer errors in position estimation, such errors also affect velocity estimates through the relation

$$v = at \qquad (2\text{-}8)$$

where again a is assumed to be constant. Using the same value for acceleration bias as above (100 μg), after 100 sec, the velocity error is given by

$$\Delta v = 0.0032 \times 100 = 0.32 \text{ ft/sec} \cong 10 \text{ cm/sec.} \qquad (2\text{-}9)$$

The meaning of this result is that if velocity could be sensed to a precision better than this value, then such measurements could be used to improve position estimates beyond that obtainable by double integration of accelerations for the accelerometer errors and integration times assumed (100 μg and 100 sec. respectively). This possibility is explored in the following paragraphs.

The three most common techniques for velocity measurement suitable for submerged vehicles are doppler sonar, correlation velocity log sonar, and water speed measurements. Each of these techniques is limited to approximately one percent of full scale and the latter is in addition subject to further error due to the effect of ocean currents. If this accuracy is applied to the vehicle characteristics assumed in this study, then

$$\Delta v = 500 \text{ cm/sec} \times 0.01 = 5 \text{ cm/sec.} \qquad (2\text{-}10)$$

This is somewhat smaller than the error in inertial measurement of velocity and shows that such data might be useful for the system parameter values assumed. However, preliminary investigations indicate that both types of sonar are too large and demand too much power to be considered for the proposed SANS system. As a consequence, only water speed sensing is considered further in this report.

An alternative to explicit water speed sensing is possible in AUVs due to the high precision available in depth measurements. This allows differentiation of such data to obtain , $\dot{z}$ which in turn allows estimation of forward velocity, v through the relation

$$\dot{z} = v \sin \theta \qquad (2\text{-}11)$$

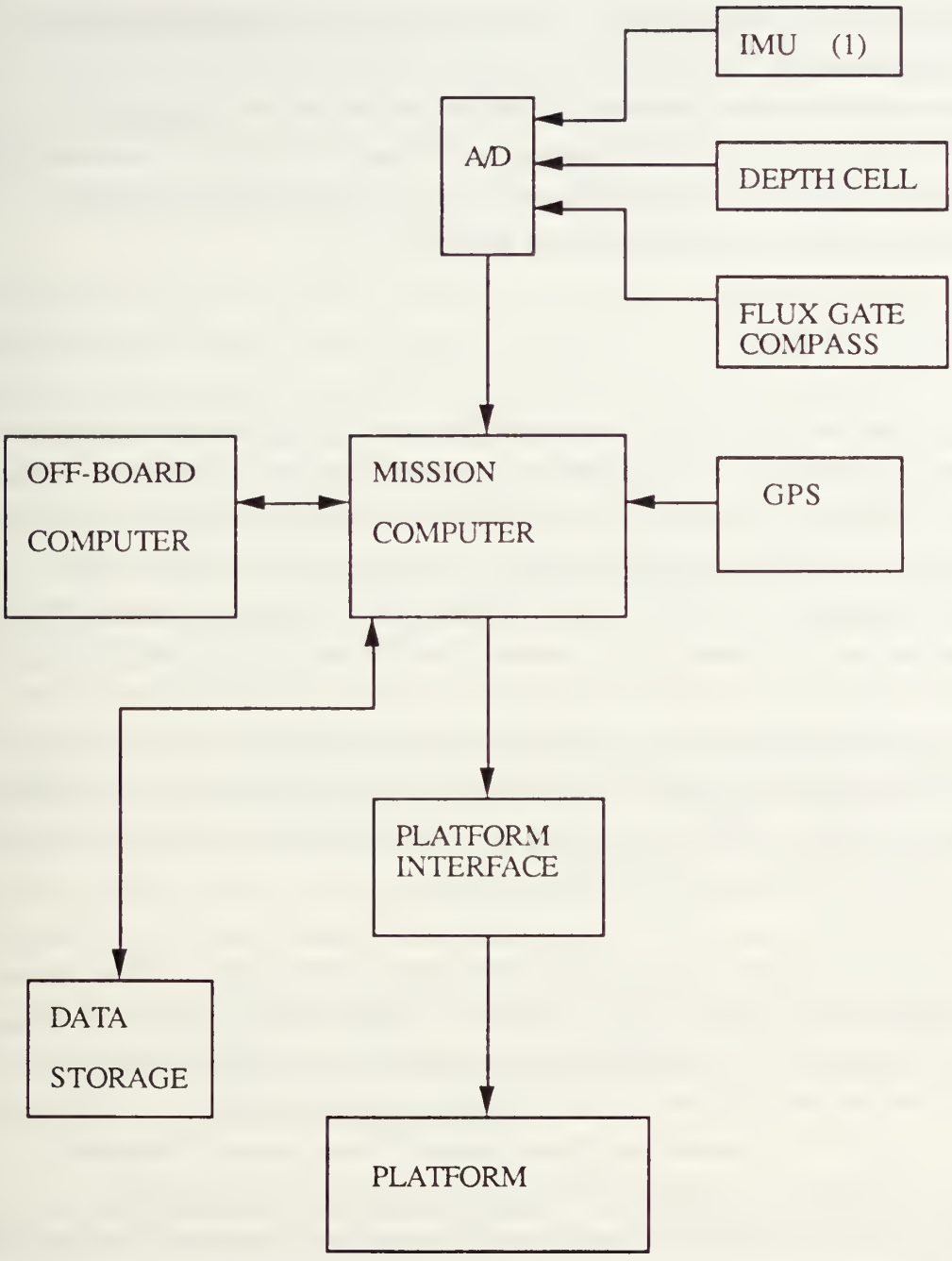
where θ is the dive angle. Solving this relation for v yields

$$v = \frac{\dot{z}}{\sin \theta} \simeq \frac{\dot{z}}{\theta} \qquad (2\text{-}12)$$

Assuming again a dive of 100 sec duration, Eq. (2-12) must produce a velocity estimate accurate to better than 10 cm/sec in order to be useful. At a speed of 5 m/sec (10 kt.) this amounts to an accuracy of two percent. This level of precision should be attainable by estimation of by means of an optimal observer [7]. In addition, assuming an average dive angle of 5 degrees in magnitude, it follows that θ must be known to an accuracy of 0.1 degrees or better. Such accuracies are consistent with the general performance parameters of Table 2-1 and indicate that there is promise in this approach to speed estimation. However, this is a complicated matter and requires an in depth study of optimal estimation techniques to determine what is actually possible [7,15]. Moreover, whatever means is used for water speed measurement, GPS position and/or velocity information will be needed in order to include the effects of ocean currents.

2.4 Proposed System

Taking into account the above information and calculations, the overall hardware configuration proposed for the SANS system is as shown in Fig. 2-1. It should be noted that a flux gate compass has been added to the sensor suite proposed. Such sensors are now available in a package occupying about 10 cu. in., including electronics, and requiring



(1) THREE ACCELEROMETERS
THREE RATE SENSORS

Figure 2-1: Hardware Block Diagram for SANS On Board Navigation and Mission Control Package

less than 1 watt in electrical power[16]. Heading accuracy of 0.5^{o} rms is advertised. It is felt that such a sensor may be useful in maintaining heading between GPS fixes as an alternative to real-time optimal estimation of position and orientation, a capability not proposed for SANS, since such computations are to be performed only by post-processing after mission completion. A water speed sensor has not been included in this figure, although further study may indicates that it is needed [6, 15].

2.5 Summary

Although the error analysis in this part of this report is greatly simplified, the overall conclusions are generally in agreement with recent experimental results relating to land navigation with state of the art strapped down IMU systems [8]. Specifically, an accuracy of 5 meters rms in position estimation after 100 seconds submerged operation seems feasible. An important feature of the post-processing position estimation scheme proposed in this report is that this accuracy can be extended both forward and backward from a GPS fix, thereby allowing at least 200 sec submerged operations between GPS fixes. With a maximum antenna exposure time of 30 secs, this means that the SANS system will require surfacing for a *maximum* of fifteen percent of the time during the precision mapping phase of its mission. However, recent experimental results obtained at NPS indicate that *average* exposure time will be much less than this figure [33].

Of course, all of the numbers relating to INS performance in this report are subject to refinement by further analysis [14,15]. However, it is believed by the authors that the feasibility of meeting the inertial navigation accuracy requirements of SANS has been established, and that the performance analysis presented is conservative; i.e., careful application of optimal estimation theory should yield still better results. Taking into account the capabilities of differential GPS presented in the next chapter, and the availability of microcomputers of extremely small size and low power demand as described in Chapter 4 of this report, we are convinced that the design objectives outlined in Chapter 1 can be met with existing technology at acceptable cost in FY94.

3. GPS SUBSYSTEM

3.1 GPS Navigation

3.1.1 GPS System Description

The Global Positioning System (GPS) is currently under development by the DoD as the next generation of space based navigation system [17]. It will consist of a set of 24 satellites in high earth orbit that have periods of almost 12 hours. The satellites broadcast information to allow ranges to be computed between a receiver on the earth and the satellites. When four or more of these ranges are observed, a user can compute his position and velocity in all 3 directions. If only 3 are observed, horizontal (2-dimensional) position and velocity can be computed. In late 1991 worldwide, 24 hour, 2-dimensional capabilities were established. In late 1992 worldwide, 24 hour, 3- dimensional capability is scheduled to be available.

This system is totally passive as far as the user is concerned. He just uses a specialized radio receiver. Today GPS receivers not only receive the signals, but also compute positions. Normally the receivers have an 8 state Kalman filter, with three position states, 3 velocity states, and two for estimating the receiver's timing error. The GPS has two levels of accuracy, the Standard Positioning Service (SPS) and the Precise Positioning Service (PPS). The former is designed for civilian use and has an accuracy level, set by policy, of 100 m horizontal accuracy. The PPS, designed mainly for the military, has a 16 m 3-dimensional accuracy and 0.1 m/s velocity accuracy [18]. Use of the PPS requires cryptographic information and cryptokeys.

The deliberate degrading of the accuracy of the GPS signals to limit the accuracy that a stand alone non-PPS user can obtain is called Selective Availability (SA). There is also a technique to prevent the spoofing of receivers with fake GPS signals. This is called Anti-Spoofing or AS. It is accomplished by encrypting the more precise ranging signals. However AS does not effect the signal used by the SPS.

3.1.2 Differential Positioning

A method to improve the accuracy of the SPS has been under development in the science and surveying community for a few years. This is differential GPS [19]. In this technique a reference receiver sits on a known location and records data and solutions. The errors in

the known position and/or data are then used to adjust the positions of receivers at unknown locations. This method has been demonstrated to work over fairly large ranges (few hundred Km) and give accuracies at the 2-4 m, 1 sigma level -- better than PPS [20,21]. This technique is currently under study for many precision applications such as aircraft landing [22,23]. It also has been used in a harsh at sea environment on the tail buoy of a towed array [24].

The use of differential positioning makes high precision solutions available without the use of classified cyrptokeys, that is within the SPS. Kremer [25] estimates that SA effects will be reduced to under 5 m at 250 km. In specific cases the effects have been measured in SA test periods to be under 10 m at 1300 km [26].

The use of differential positioning requires that the data or solutions from the unknown location and the reference station be processed together. This can be done in real time through communication channels or in post processing. In the case of an autonomous underwater vehicle application, the latter would be used. An example of differential errors from these receivers is shown in Fig.3.1 [21]. In all cases the receiver produced errors under 10m, including the Magnovox MX4200. This is one of the type of receivers under consideration for this project.

3.1.3 Message Data

In order to determine where one is from ranges to any location, including satellites, one must know where the other end of the range measurement is at -- ie where the satellites are. This is accomplished in GPS by the addition of some low data rate information to the ranging signal [18]. Each satellite broadcasts a high precision version of its own location via parameters for a mathematical model. This is called the broadcast ephemeris. This is repeatedly broadcast every 30 sec. It is good for high precision navigation for about 6 hours. In a differential mode it could be used significantly longer.

In addition to ephemeris data, the satellites broadcast information about the locations of all operating GPS satellites. This is called the almanac. It is broadcast at a lower rate and takes 12.5 min. to cycle. However it is valid for about a month and is changed only weekly. Its main purpose is to give receivers a knowledge of what satellites are visible and aid in acquisition of signals.

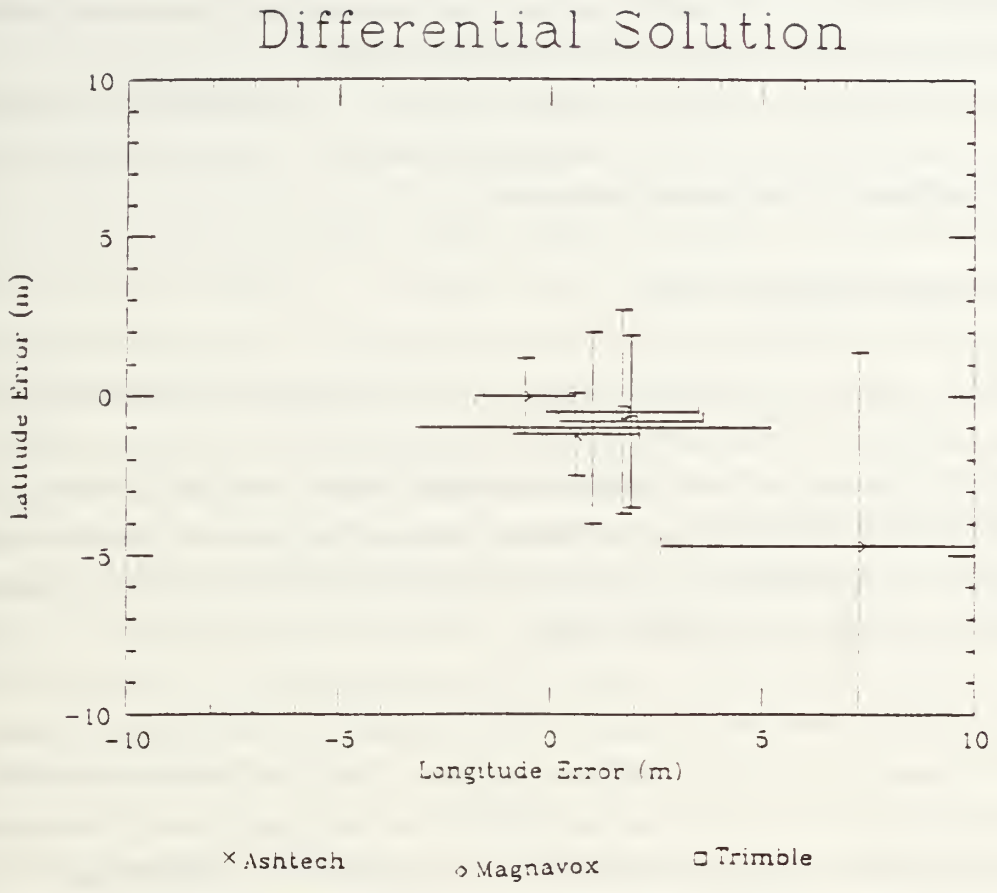


Figure 3-1: Horizontal Diffential Solution Accuracy One Second Data Averaged Over Three Hour Mission [21]

3.2 Mission Specific Factors

The general idea here is to have a GPS receiver on an Autonomous Underwater Vehicle which will occasionally get a fix to update an inertial system. The main requirement is for the acquisition of data for post processing. There is a secondary goal to obtain real time navigation solutions for navigation to and from the mission area. Key factors in this system will be exposure time, size, weight, and power. The exposure time is considered critical and will be the main driver on equipment selection. (There are several receivers that are essentially equivalent on the other factors.)

3.2.1 Signal Acquisition Times

In order to acquire a satellite signal, a GPS receiver must find it in a two parameter space -- range and Doppler. It can do this if it knows roughly where it is, where the satellites are, and what the state of it's clock is (time and frequency error) . Having an almanac will normally satisfy the requirement for knowledge of satellite positions. The knowledge of its own location must be known only to a hundred Km or so. Neither of these should be difficult for the mission type under study.

The state of the receiver's clock may be a more difficult item. Several vendors believe that this will be the limiting factor. While re-acquisition times of a few seconds are quoted and demonstrated with off times of a minute or less by several vendors, after a few minutes times up to 4 min. are quoted. Fortunately one vendor (SEL) has demonstrated a proprietary scheme that keeps the acquisition time for 3 satellites at 30 sec or less even for 8 min. of off time[33].

3.2.2 Antenna

A key consideration for the GPS subsystem will be the antenna. GPS signals cannot be received through water. Even a cm of water will attenuate the signal to a useless level. Therefore the antenna will have to clear the sea surface to be used. It does not have to raise above the surface.

There are three manufacturers of conformal antennas. These are generally hemispherical in coverage and are on a disk about 3 in. in diameter. This disk is normally flat but can be manufactured in a curve. Two or more antenna's can be connected together to give better

angular coverage in an environment where the vehicle may be rolling. This could make for a good low-drag package.
Discussions with two of the three antenna manufactures have established that a such a low profile antenna can be produced. One (Ball Aerospace) has even produced an antenna of this type for an underwater application.

3.2.3 Satellite Availability

In order to fulfill the mission of accurate post processed positioning for the AUV, the GPS receiver will have to obtain data from at least 3 different satellites on any given exposure. After late 1992 there should be 4 - 7 satellites always visible at an elevation angle of 10 deg or more anywhere anytime. The fact that the antenna is at the sea-surface, ie on the geoid, can be used to strengthen the solution.

The acquisition of data from 3 satellites does not guarantee that the receiver will produce a real time solution. However some receivers can be set to a height hold mode to force them to produce a solution on 3 satellites. This would aid in the secondary goal of real time navigation. It should be borne in mind that any real time SPS solution will be in error by the SPS error limit. (The reference station will, of course, remove the SA error.)

3.3 Hardware Availability

Small, low power SPS GPS receivers and "engines" are now coming onto the market. Engines are single boards with a full GPS receiver that acquires signals, tracks satellites, produces navigation solutions, and outputs solutions and data. They typically do not have a power supply or any control device. Interface is usually via RS 232 ports.
Appropriate systems are for sale from SEL, Rockwell-Collins, Magellian, Trimble, Ashtech, and others. They all have power requirements in the 3-7 W category. Most are provided as single boards less than 4 " x 6 " in size. Solution and data at a 1 Hz or sometimes 2 Hz rate are available.

These engines are derivatives of small receivers that are slightly bigger and usually have the same interfaces. Two of these small complete receivers have been examined. The Magnavox MX 4200 has been extensively studied in the Mapping, Charting and Geodesy program at NPS. It has typical acquisition times of 10 - 12 seconds with dead times of 30 sec. A demonstration of the SEL system was held at the vendor site in Los Angeles. It had

excellent acquisition characteristics, acquiring 4 satellites in 7 seconds and producing a solution after having been fully powered off for 30 min. A more detailed statistical analysis of this receiver can be found in [33].

3.4 Risk

It is not felt that there is much risk in the GPS subsystem. Hardware which meets the need has been found from multiple vendors. Cost is in the $ 1 K range for the bare engines. There will need to be some work done on choosing a specific system, designing the interface to the mission computer, and especially the antenna sub-system. This should not be a problem and it is expected that a prototype for bench testing could be available in July 92.

4. COMPUTER HARDWARE AND SOFTWARE ARCHITECTURE

4.1 SANS Computer Hardware

The computer hardware for the proposed system will be a single processor system. A single processor system has advantages over a multi-processor system because it uses less power and occupies less space. Among currently available processors, either the Intel 80386 (or higher powered processors in the Intel processor family) or the Motorola 68020 (or higher powered processors in the Motorola processor family) will be good candidates because of their proven performance and popularity.

Although there are many small single board computers on the market based on these processors, there is little difference in their performance, size, and power requirements. A typical configuration of a single board computer is one CPU, two or three interface logic LSIs, and memory ICs. Thus, their board sizes are roughly the same, and their power consumption figures are approximately equivalent if the same type of power conservation chips are used. Because of these considerations, in this section, two typical small board-type computers, which are selected from each processor family, will be discussed. They are Pro-Log's 386SX/AT Software Compatible Single Board Computer 7872, and Onset Computer's Tattletale Model 7. The former computer has a 16MHz Intel 80386SX CPU, and the latter comes with a 16MHz Motorola 68332 CPU, which is a one chip computer with functionality equivalent to a Motorola 68020 with internal memory and I/O.

The major advantage of the former computer is its compatibility with an IBM AT which is the most widely used computer. Thus, there is a great deal of software support readily available. For this reason, and in particular because good Ada compilers are now available for 80386 based systems, in our development of prototype software we have temporarily assumed that the Pro-Log 7872 will be selected for SANS. This decision may, of course, be altered after further study.

The major advantage of the Tattletale Model 7 board computer is low power consumption and an adjustable system clock from 320KHz to 16 MHz. This board's power consumption is very low compared with the Pro-Log 7872 computer because the specific CPU is a low power system designed for automotive control applications. (There is more than a factor of 10 difference in power requirement between a normal CPU and a low-power CPU [27].) For example, at 16MHz system clock speed, it consumes less than one

tenth of that of the Pro-Log 7872 system. Moreover, the latter system can even save more power by lowering the clock speed if all of the 16MHz processing power is not required at a given time for the proposed system. The reason is that power consumption of a micro-computer system is roughly proportional to its system clock speed. On the other hand, the latter system requires a development environment or a development system because it is not compatible with any existing stand alone computer. Specifically, the latter system utilizes a Macintosh computer as its development environment, utilizing the THINK C compiler and its associated CrossCut development software. THINK C is one of the most popular C compilers for a Macintosh computer. Currently, Tattletale's development environment does not support the Ada programming language. We view this as a very serious limitation. Important characteristics of the two systems discussed here are summarized in Table 4-1. It should be noted that these two system were selected merely as representatives of two quite different solutions to the real time computer needs of SANS, and that neither necessarily represents recommended choice for implementation.

Table 4-1: Comparison of two on-board computer hardware systems

Computer System	Power Consumption	Size	Bus	Programming Languages	Development Environment
Pro-Log 386SX/AT (7872)	5.1W @ 16MHz	4 X 6 X 1 (24 cu in)	STD Bus	Ada, C++, C	Standard IBM-PC/AT Compatible*
Onset Tattletale Model 7	0.35W @ 16MHz 0.15W @ 10MHz	4 X 2.75 X 1 (11 cu in)	None (Connectors)	C, TXBASIC	Macintosh with special software & hardware**

* : 7872 is IBM-AT compatible. No special development system is required for software development.

**: Model 7 is not Macintosh compatible. It needs a development system which is composed of a Macintosh computer , CrossCut development software, and a Mac Parallel I/O board.

4.2 SANS Software Architecture

The software, which will be operated in the Small AUV Navigation System based on one of the above mentioned or equivalent on-board computers, will be developed according to a structured software engineering concept. Thus, the software will be composed of hierarchically organized program modules which have clean module interfaces and clear functionality. When a first version of the proposed software is completed, it will be easily upgraded for a newer version. Moreover, some of the modules can be re-usable for a similar system. As matter of a fact, we have developed the prototype software listed in Appendix 1 partially with the support of another project concerned with a surface ship navigation data log (SNDL) [28] since there is a large degree of commonality of this system with SANS.

The proposed software system is shown in Fig. 4-1. This figure uses Yourdon's notation [29] in which a circle (or a bubble) represents a system or a sub-system (process), and a square box, called "terminator", shows an external agent interacting with the system. In addition, the double lines designate a file on a computer disk or in non-volatile memory, and arrows show data movement. A label on a arrow explains the contents moving through the arrow. As shown in Fig. 4-1, the Small AUV Navigation System gets data from various sensors; i.e., a GPS receiver, an IMU system, a depth cell, and a flux gate compass. It also gets a mission description from a mission file. In turn, it generates on-line control commands for the small AUV and two files, GPS raw data log and navigation data log, for post processing purposes. The former file is a simple data log storing incoming data from a GPS receiver, and the latter is processed on-line navigation data developed during mission execution. Consequently, Fig. 4-1 shows the system boundary and its interaction and is, therefore, known as a "context diagram".

The internals of the middle bubble in Fig. 4-1 is shown in Fig. 4-2. There are four bubbles in this diagram which show the overall internal operation of the system. Each bubble also has its unique bubble number. This number is used to identify each bubble and is further used as a reference number for future refinement of a bubble. For example, Bubble 2 is decomposed into four sub-bubbles in Fig. 4-3, named 2.1, 2.2, 2.3, and 2.4. The overall operation of this system is as follows. The sensor data from various sensors will be processed in the Process INS Data bubble, and the processed INS data will be given to Bubble 3 [3]. Bubble 3 will merge this data with incoming GPS data from Bubble 1 which processes GPS raw data to produce GPS position data. The corrected INS data

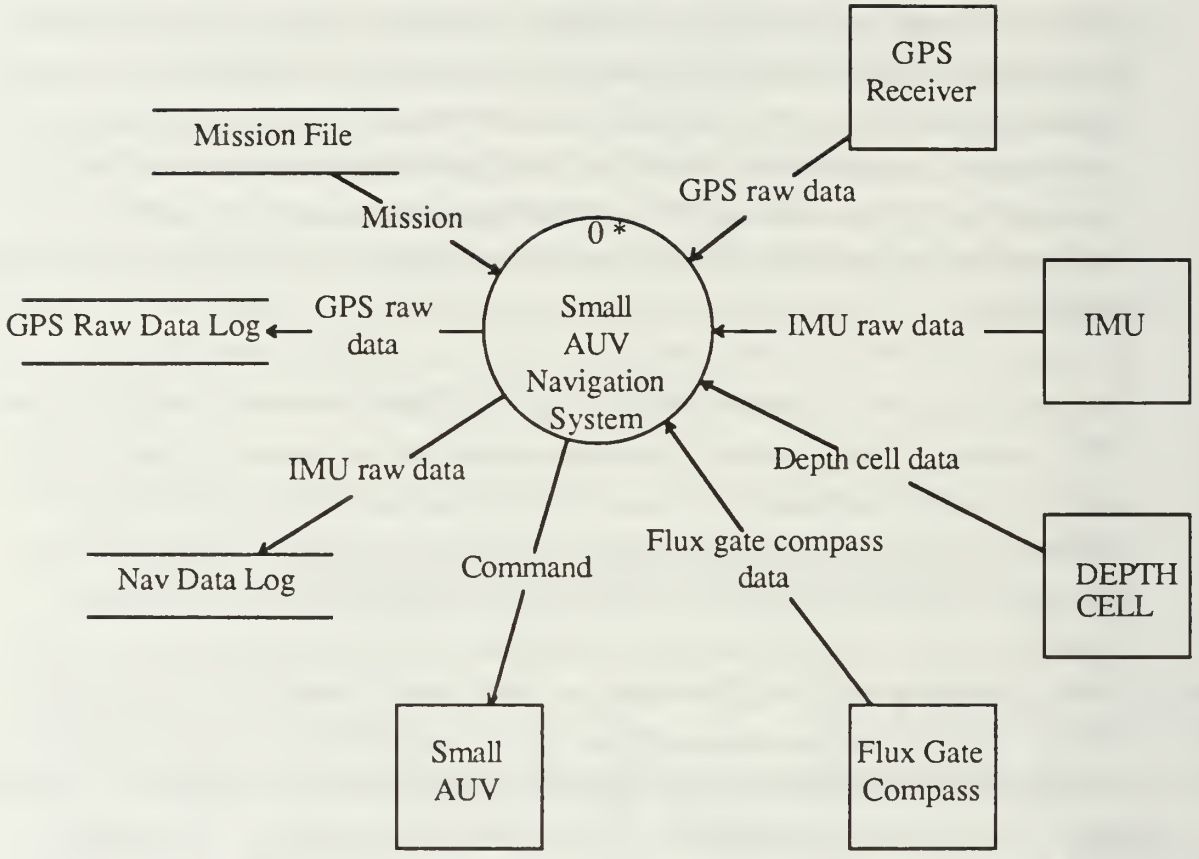


Figure 4-1: Level Top Diagram for SANS On Board Software

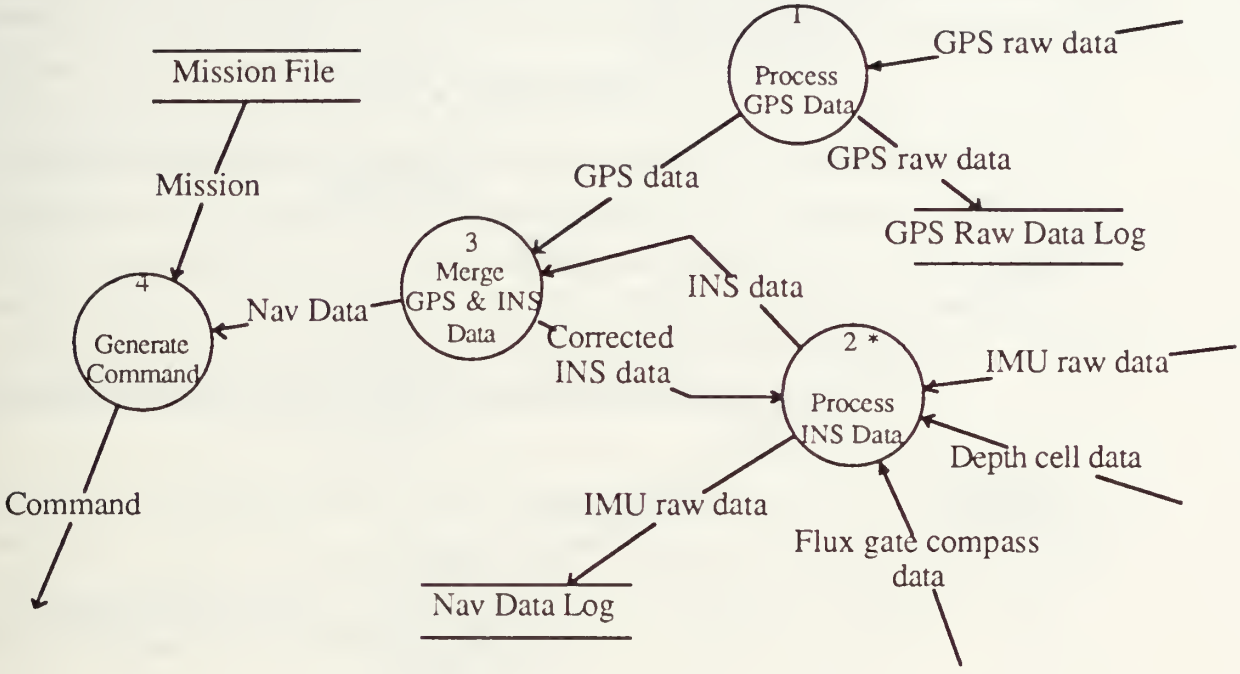


Figure 4-2: Level 0 Digram for SANS On Board Software

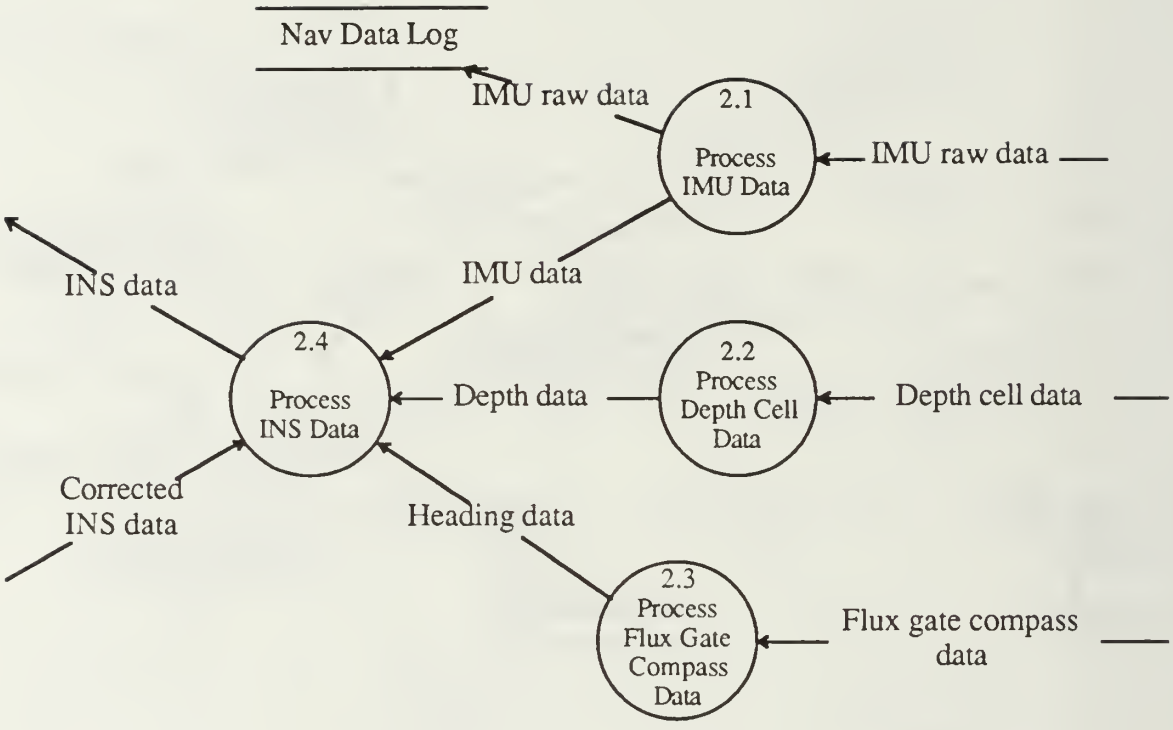


Figure 4-3: Level 2 Diagram for SANS On Board Software

based on GPS data by Bubble 3 will be returned to Bubble 2 to correct INS drift. When real time navigation is required, Bubble 3 also generates navigation data to produce Nav Data Log. The navigation data is used by Bubble 4 to generate on-line control commands in conjunction with the mission description from the Mission File. Although details of Bubble 2,3, and 4 are not finalized, the details of Bubble 1 are basically available because it can be directly borrowed from the SNDL system. Only very small modifications for a new interface to the new bubble are expected. The completely debugged and working Ada code is presented in the attached Appendix.

The details of Bubble 2 in Fig. 4-2 are presented in Fig. 4-3. Though IMU data is minimally required to produce INS data, heading data from a flux gate compass is also used to obtain better performance. Additionally, depth data is utilized to improve INS data accuracy for an AUV which follows a wave-like periodic movement in a vertical plane. In this scheme, a depth change is used to calculate the forward speed of the AUV. The above mentioned sensor data fusion will be performed by Bubble 2.4.

It is not clear at the present time whether it will be possible or not to accomplish real-time estimation of position using optimal filtering techniques to combine IMU data with GPS and other sensor data. However, such processing will definitely be required in post-mission data analysis as discussed in the following section of this report.

4.3 Mission Planning and Post-Mission Data Analysis Hardware and Software

A Sun Sparc based laptop computer or a Intel 80386(or 80486) based notebook will be employed for mission planning and post-mission data analysis to meet the following objectives:

a. High processing power requirement due to complex data manipulation.
b. Portability of computing hardware.
c. Low price.
d. Ease of operation.
d. Volume, weight, and power constraints are not critical.

Planning a mission composed of three phases, i.e., transit, mapping, and return, may be not a trivial task. During this stage, a path,which satisfies all the mission requirements from a launching point to destination should be devised while considering a complex operational environment. Though this task usually requires a human expert, it can be

automated by a mission planning expert system so that any one of operational crew, even though he has no expertise on the specific mission planning, can plan out a mission. Currently, a similar mission planning system [2] for the NPS AUV II is running on a Sun Sparc workstation and is written in KEE [30] and Common LISP. Thus, it is expected that the mission planning part will be written in a similar artificial intelligence programming language, such CLIPS [31], KEE, LISP, or Prolog.

When mission execution is completed, the data collected by a SANS package, which was carried by a small AUV, is uploaded to the same portable computer. Then, the collected data, which is composed of GPS data and IMU data, will be processed by post-mission analysis software. In this stage, much higher positional and velocity precision than on-line navigation data provided by a SANS package will be achieved. Specifically, differential GPS will provide precise positional and velocity information, and this data will be used to correct IMU drift errors so that a precise mission trajectory executed by an AUV can be obtained. These process will demand substantial computing power for data fusion and data smoothing utilizing Kalman filters. Like the on-board part of SANS, this part of the software will be developed according to a structured software engineering concept for ease of maintenance and upgrade in the future.

5. EXAMPLE SYSTEM DESIGNS

As a result of this study, two preliminary designs were developed: a baseline design and an interim design. The baseline design is expected to meet all operational requirements outlined at the beginning of this report at a hardware cost not exceeding $50,000 in fiscal year 1994 dollars. The interim design represents a decrease in capability, but can breadboarded in fiscal year 1992 and a lower cost. It is important to realize that these systems are not mutually exclusive. In fact, any development and evaluation done on the interim design will be directly employable to the final development of the baseline design.

The interim design is shown below:

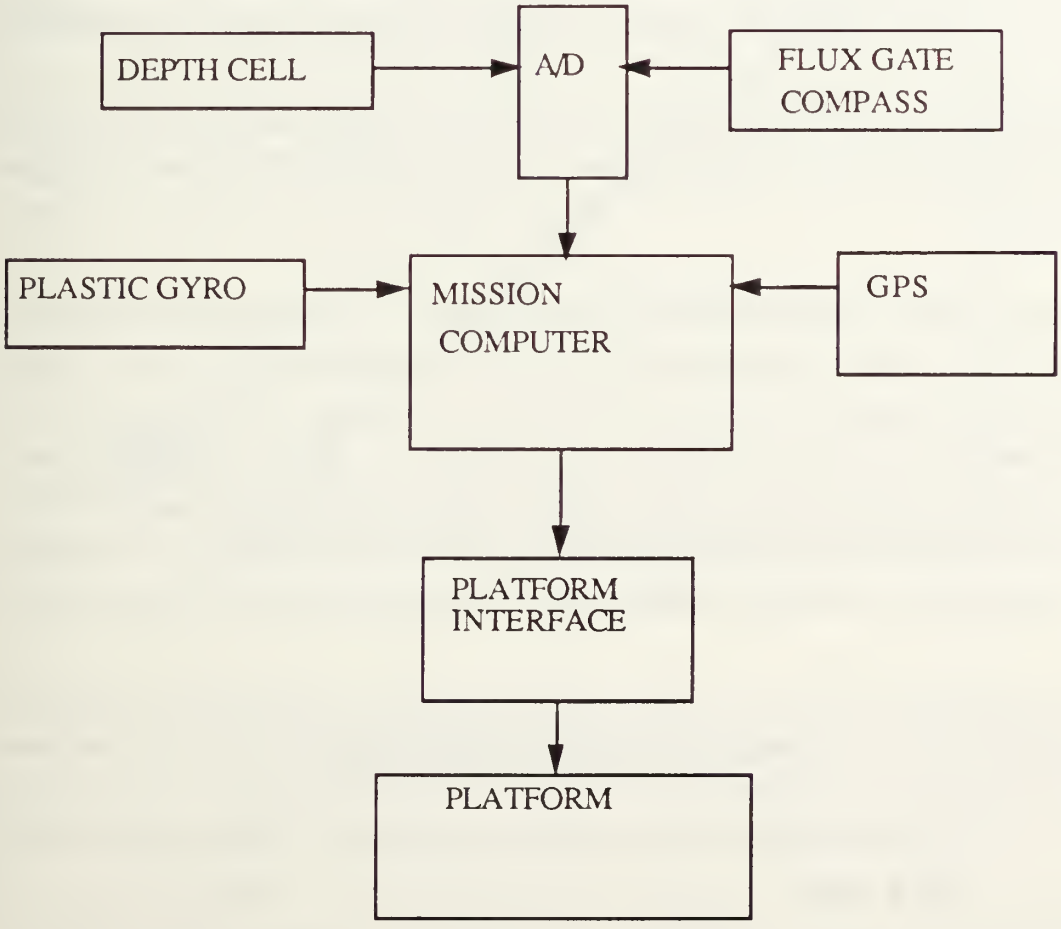


Figure 5-1: Interim Design

The primary difference between the two designs revolves around the inertial navigational part. In the interim design, a plastic gyro developed by Gyration Corporation [34] will be used to provide a vertical reference to a few degrees of error for shallow water applications. Under the interim design concept, the AUV will have to surface every time an object of interest is found in order to use GPS information for object location. As shown in Fig. 5-2, Horizontal displacement from where the vehicle surfaced to the actually location of the object is computed by

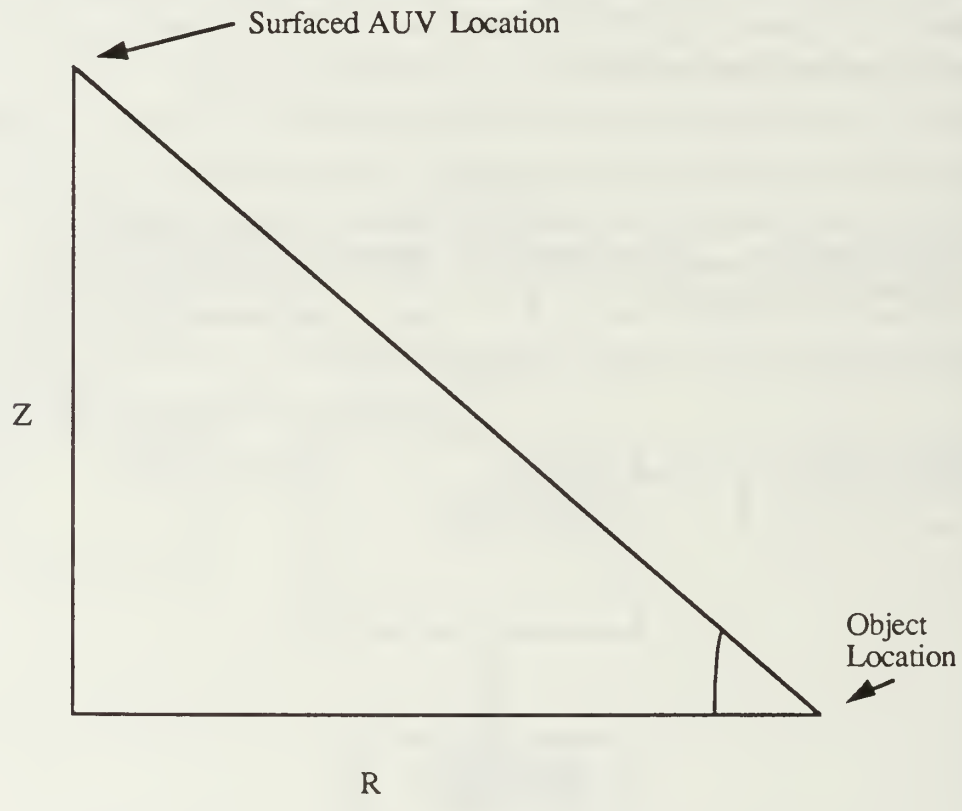


Figure 5-2: Object Location Diagram

$$Z / R = \tan\theta \qquad (5-1)$$

where, Z is the depth, R the horizontal displacement, and θ the angle of ascent. The depth is provided by the depth cell, and the angle of ascent is averaged from readings provided by the plastic vertical gyro.

Solving for R,

$$R = Z / \tan\theta \qquad (5\text{-}2)$$

The direction of the location of the object of interest from the surfaced vehicle is calculated by:

$$\Delta EAST = R \sin\psi \qquad (5\text{-}3)$$

and

$$\Delta NORTH = R \cos\psi \qquad (5\text{-}4)$$

where ψ is the average heading provided by the compass during the ascent.

Of course this analysis is based on an assumption of an essentially constant AUV climb angle. More realistically, these relationships could be applied differentially and the result summed to obtain east and north displacements.

The mission computer proposed is an Octagon Systems Corporation 5012 PC Control Card. This is an 80C086 board rated at 4.7 or 12 mhz. This board carries up to 2 MB of DRAM and is EPROM capable. It runs DOS 3.31 and has a watchdog timer that will reset upon a program crash. This is software enabled. The clock speed may be slowed down to conserve power during periods of light operational loads. The pre-mission program and the operating system DOS 3.31 would be loaded into the EPROM. The 2 MB of DRAM are believed to provide more than enough memory for data storage during the mission.

The proposed depth cell is a pressure transducer by Celesco. Celesco manufactures the transducer currently in use in the NPS Model II AUV. The transducer for this application is the DP30 with a range of +/- 0.1 to +/- 500 psi. The linearity is +/- 0.5% of full scale. Celesco also provides a compatible low cost demodulator for the transducer.

The GPS receiver suggested is the GLOBOS LN 2000 GPS engine by SEL. Extensive test and evaluation was done on this receiver [33]. The satellite acquisition times and times to a solution as well as the stability of the solutions are all minimally acceptable for the interim

design. The test and evaluation of this receiver clearly demonstrates that off the shelf civilian use receivers will be adequate for this project.

The preferred magnetic compass is manufactured by KVH Industries and is the C100 Multi-Purpose Digital Compass. This compass has an accuracy of +/- 0.5 degrees and has an auto-calibration method that deals with the magnetic signature of the host platform.

The proposed A/D converter is manufactured by the same company that produces the mission computer. The advantage to this selection is that the mission computer and A/D converter are compatible and would both fit in the same card cage if used. The A/D converter of choice is the 5710, a high resolution, low cost analog card.

Not shown on the interim design diagram are the associated I/O interfaces and the power supply regulator. The interfaces are standard RS232 interfaces. The power supply regulator will provide a constant power source to the components at about 85 percent efficiency and would be able to handle the requirements for all components. It is also computer controlled and may be used to turn the different components on and off as needed lowering power consumption.

The final two concerns of the interim design relate to the battery pack and to the size of the system as a whole. The battery supplied power may be of two types: rechargeable batteries or throw away batteries. Initial estimation of energy requirements in a worst case analysis based on a 24 hour mission predicts a total of 180 watt hours with an average power requirement of 7.5 watts and a peak of 17.5 watts. The best solutions include alkaline batteries which provide 30-45 watt hours per pound at 2-3 cubic inches per pound or extra life zinc chloride with some recharge capability at 40 watt hours and 3 cubic inches per pound. An important consideration is for the power package is to be neutrally buoyant, which places a limit of around 5 lbs of batteries. By using a two package system, one for the batteries and the other for the SANS system, the best configuration would be to have each of the individual packages neutrally buoyant.

The baseline design, which is very similar to the interim design, is shown below:

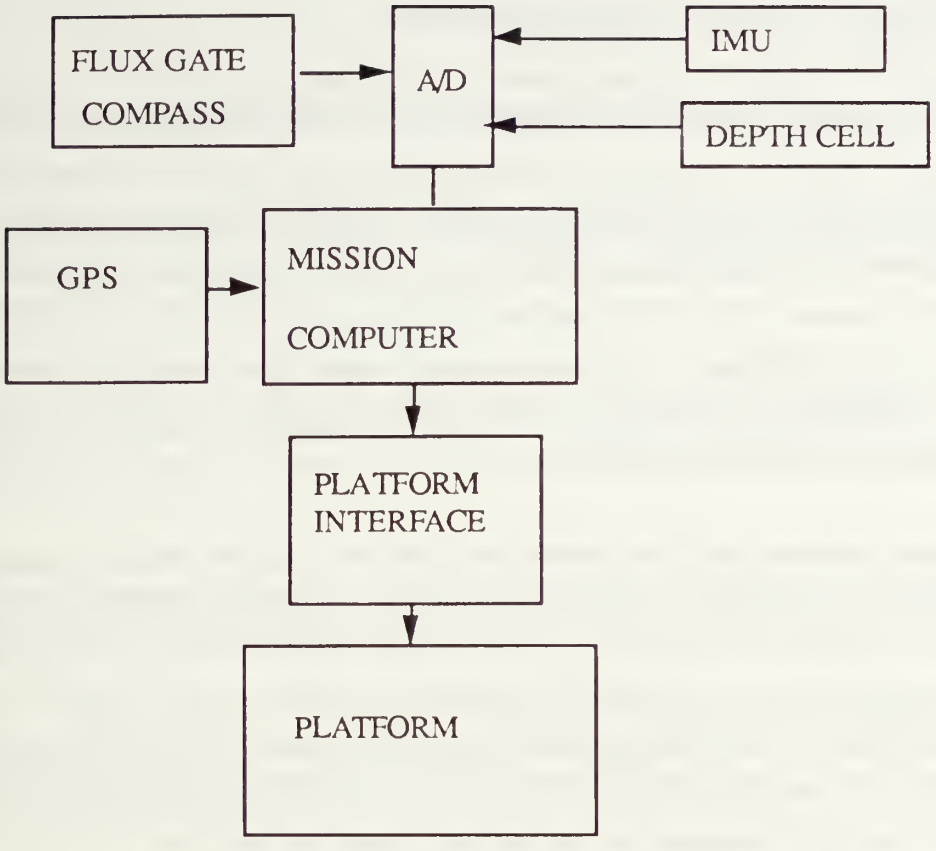


Figure 5-3: Baseline Design

The primary difference between the two design as stated before is the navigational part. In the baseline design the IMU will consist of three accelerometers and three rate sensors. The LN-200 IMU being developed by Litton will be available in late 1992 or early 1993. The angular rate bias of this unit is 1 degree per hour, and would allow for the AUV to stay submerged for 5 minutes or longer.

The rest of the baseline design, size, and power requirements are as outlined in the interim design. The different technologies of the varying components changes literally on a week to week basis. The system described above represents the components that would be used today. All of these components are off the self and require little or no modification.

Further details relating to the both the interim and the baseline design will are provided in [14].

6. SUMMARY AND RECOMMENDATIONS

The authors believe that the information presented in this report justifies the conclusion that an interim SANS system employing a pop-up maneuver for location of submerged objects could be constructed in breadboard form in FY92 at a hardware cost not exceeding $25K. A draft proposal outlining such an effort has been recently submitted to NOSC Hawaii[32]. We further believe that the FY92 breadboard could be repacked for at sea testing in FY93. Finally, it appears that a prototype (baseline) system meeting all mission requirements could be assembled in FY94 at a hardware cost of approximately $50K. We are interested in participating in all of these efforts.

We have not submitted any cost breakdowns for either the interim or baseline system because we have no actual quotations for any of the components. Rather, our overall cost estimates are derived from visits to vendors and many hours of telephone conversations with both technical personnel and sales representatives for the various vendors potentially involved. However, based on our collective experience with the NPS AUV and related systems, we are convinced that our cost estimates are conservative. Should NOSC require that we obtain actual quotations before proceeding with the next phase of this project, we would of course be willing to include such data in our proposal.

Because this phase of this investigation was limited to a feasibility study, we have not accomplished a detailed design of either the interim or baseline system concepts. Rather, the two designs described in this report are intended to be illustrative. If the work proposed in [32] is funded, we would of course complete, construct, and bench test the interim system. Full funding of this proposal would also permit detailed design and performance analysis for a baseline system intended to meet all mission objectives. In order to ensure timely development of a prototype SANS system suitable for at sea testing, we strongly recommend that, at a minimum, funds for bench testing of interim system components be provided in FY92.

References

1. McGhee, R.B., *Technology Survey and Preliminary Design of a Small AUV Navigation System*, Research Proposal, Naval Postgraduate School, Monterey, CA, 13 February, 1991.

2. Kwak, S.H., Ong, S.M., and McGhee, R.B., "A Mission Planning Expert System for an Autonomous Underwater Vehicle", *Proceedings of the Symposium on Autonomous Underwater Vehicle Technology*, IEEE Oceanic Engineering Society, June 4-6, 1990, Washington DC., pp. 123-128.

3. Healey, A.J., McGhee, R.B., Cristi, R., Papoulias, F.A., Kwak, S.H., Kanayama, Y., and Lee, Y., "Mission Planning, Execution, and Data Analysis for the NPS AUV II Autonomous Underwater Vehicle", *Proc. of First IARP Workshop on Mobile Robots for Subsea Environments*, Monterey Bay Aquarium, Monterey, California, October 23-26, 1990, pp. 177-186.

4. Zyda, M.J., McGhee, R. B., Kwak, S.H., Nordman, D.B., Rogers, R.C., and Marco, D., "Three-Dimensional Visualization of Mission Planning and Control for the NPS Autonomous Underwater Vehicle", *IEEE Journal of Oceanic Engineering*, Vol. 15, No. 3, July, 1990.

5. Floyd, C., Kanayama, Y., and Magrino, C., "Underwater Obstacle Recognition Using a Low-Resolution Sonar", *Proc. of 7th International Symposium on Unmanned Untethered Submersible Technology*, New England Center, Durham, New Hampshire, September 23-25, 1991.

6. Clynch, J. R., et al., "Monterey Bay Precise Positioning Experiment: Comparisons of GPS Receiver Solutions Under Dynamic Conditions", *Proc. ION-GPS-91*, Albuquerque, New Mexico, September 1991.

7. Nagengast, S., *Correction of Inertial Measurements Using GPS Update for Underwater Navigation*, M.S. Thesis, Naval Postgraduate School, Monterey, CA., March, 1992.

8. Gaskill, S. K., "Honeywell Avionics--Commonality Across the UAV Family", presented at *Annual Meeting of Association for Unmanned Vehicle Systems*, Washington, DC., August, 1991.

9. Anon., *LN-200 Solid State Inertial Measurement Unit*, Litton Guidance and Control Systems, Inc., 5500 Canoga Avenue, Woodland Hills, CA.

10. Adams, G. W., Honeywell, Inc., private communication, 4 June 1991.

11. Johnson R., Honeywell, Inc., private communication, 14 August 1991.

12. Felsing, G. W., Systron Donner, Inc., private communication, 13 June 1991.

13. Anon., New Angular Rate Dynamics Instrumentation and Circuits, Humphrey, Inc., 1991.

14. McKeon, J. B., *Integration of GPS/INS Capability Into a Small AUV*, M.S. Thesis, Naval Postgraduate School, Monterey, CA., March, 1992.

15. Miller, C., *Application of Extended Kalman Filter to a Model-Based Navigator for an AUV*, M.S. Thesis, Naval Postgraduate School, Monterey, CA, December, 1991.

16. Anon., *KVH C100 Multi-Purpose Digital Compass Sensor Preliminary Data Sheet*, Middletown, Rhode Island, 1991.

17. Milliken, R.J., and Zoller, C.J., "Principles of Operation of NAVSTAR and System Characteristics", *Navigation Special Issue: Global Positioning System , Vol I*, Institute of Navigation, 1980, p 3.

18. Spilker, J.J., "GPS Signal Structure and Performance Characteristics", *Navigation Special Issue: Global Positioning System, Vol I*, Institute of Navigation, 1980, p 29.

19. Kalafus, R.M., Vilcaus, J., and Knable, N., "Differential Operation of NAVSTAR GPS", *Navigations Special Issue: Global Positioning System , Vol II*, Institute of Navigation, 1984, p 197.

20. Spalching, L., Krommes, S., and Pietraszawski, D., "Status of Prototype USCG DGPS Broadcodes", *Proc. Jan-GPS-91*, Albuquerque, NM, September 4-13, 1991.

21. Clynch, J.R. et al, "Monterey Bay Precise Positioning Experiment: Comparisons of GPS Receiver Solutions Under Dynamic Conditions", *Proc. ION-GPS-91*, Albuquerque, NM September 11-13, 1991.

22. Van Driel, N., and Krijn, R., "Results of a Test Program for the Use of Differential GPS for Approach Guidance" *Proc. ION-GPS-90*, Colorado Springs CO, Sept. 19-21 1990, p 621.

23. Saunder, P.E. et al, "Results of a GPS Shuttle Training Aircraft Flight Test", *Proc. ION-GPS-91*, Albuquerque, NM, September 11-13, 1991.

24. Mo, C., H. Raemasma, and I.B. Torkildsen, "Use of GPS in Seismic Tailbuoy Tracking Systems in the North and Barents Sea", *Proc. ION-GPS-91*, Albuquerque, NM, September 11-13, 1991.

25. Kremer et. al, "The Effect of Selective Availability on Differential GPS Corrections" *Navigation*, 37, 39, 1990.

26. Tolman,B, Clynch, J.R., Coco, D.S., and Leach, M.P., "The Effect of Selective Availability and Differential GPS Positioning", *Proc. ION-GPS-90*, Colorado Springs CO, Sept. 19-21 1990, p 579.

27. Anon., *Microprocessor and Peripheral Handbook*, Intel Literature Sale, Santa Clara, CA, 1988.

28. Chin, Y.C., *The Navigation Data Logger for a Suitcase Navigation System*, M.S. Thesis, Naval Postgraduate School, Monterey, CA, June, 1991.

29. Yourdon, E., *Modern Structured Analysis*, Yourdon Press, Englewood Cliffs, New Jersey, 1989.

30. Anon., *KEE Software Development System User's Manual, Ver. 3.0*, Intellicorp, Mountain View, CA, March 1986.

31. Giaratano, J. C., *CLIPS User's Guide, Vol 1,2*, NASA, Lyndon B. Johnson Space Center, Information Systems Directorate, Software Technology Branch, 1990.

32. McGhee, R.B., *Development and Evaluation of a Prototype Small AUV Navigation System (SANS)*, Draft proposal dated 11 November 1991, Naval Postgraduate School, Monterey, CA 93943.

33. Kwak, S.H., McKeon, J.B., Clynch, J.R., and McGhee, R.B., "Incorporation of Global Positioning System into Autonomous Underwater Vehicle Navigation", will be appeared in *Proceedings of the Symposium on Autonomous Underwater Vehicle Technology*, IEEE Oceanic Engineering Society, June 2-3, 1992, Washington DC.

34. Anon., *Gyration, Incorporated: An Overview of the Company, Technology, and Products, Ver. 1.2*, Gyration, Incorporated, Saratoga, CA, October, 1991.

Appendix

Ada and Assembly Code for Processing GPS Data

```
--*****************************************************************
--
--     File Name : SERIAL_D.A
--     Author : Se-Hung Kwak
--     DATE   : 9/11/91
--
--*****************************************************************

package SERIAL is
  procedure READ_CHAR(CH, DATA_READY : out CHARACTER);
    pragma INTERFACE(assembly, read_char);

  function WRITE_CHAR(CH: in CHARACTER) return CHARACTER;
    pragma INTERFACE(assembly, write_char);


  procedure OPEN_SERIAL(PORT, BAUD, DATA_BIT: INTEGER;
                  PARITY:CHARACTER; STOP: INTEGER);


  procedure INIT_SERIAL(RX_REG,TX_REG,INT_EN,LINE_CRT,MODEM_CRT,LINE_STAT,
                  BAUD_LSB,LINE,INT_MASK,INT_NUM: in INTEGER);
    pragma INTERFACE(assembly, init_serial);

  procedure CLOSE_SERIAL;
    pragma INTERFACE(assembly, close_serial);

end SERIAL;
```

```
--*****************************************************************
--
--    File Name : SERIAL.A
--    Author : Se-Hung Kwak
--    DATE   : 9/11/91
--
--*****************************************************************

package body SERIAL is

  procedure OPEN_SERIAL(PORT, BAUD, DATA_BIT: INTEGER;
                  PARITY:CHARACTER; STOP: INTEGER) is
    OFFSET : constant INTEGER := 16#100#;
    RX_REG : INTEGER := 16#2F8#;
    TX_REG : INTEGER := 16#2F8#;
    INT_EN : INTEGER := 16#2F9#;
    LINE_CRT  : INTEGER := 16#2FB#;
    MODEM_CRT : INTEGER := 16#2FC#;
    LINE_STAT : INTEGER := 16#2FD#;
    BAUD_LSB, LINE, INT_MASK, INT_NUM : INTEGER;
  begin
    if PORT = 1 then
      RX_REG := RX_REG + OFFSET;
      TX_REG := TX_REG + OFFSET;
      INT_EN := INT_EN + OFFSET;
      LINE_CRT  := LINE_CRT + OFFSET;
      MODEM_CRT := MODEM_CRT + OFFSET;
      LINE_STAT := LINE_STAT + OFFSET;
    end if;                              -- default port2

    if BAUD = 1200 then
      BAUD_LSB := 96;
    elsif BAUD = 2400 then
      BAUD_LSB := 48;
    elsif BAUD = 4800 then
      BAUD_LSB := 24;
    else
      BAUD_LSB := 12;                 -- default port2
    end if;

    if DATA_BIT = 5 then
      LINE := 0;
    elsif DATA_BIT = 6 then
      LINE := 1;
    elsif DATA_BIT = 7 then
      LINE := 2;
    else
      LINE := 3;                     -- default 8 data bits
    end if;

    if STOP = 2 then
      LINE := LINE + 16#4#;
    end if;                          -- default 1 stop bit

    if PARITY /= 'N' then
      LINE := LINE + 16#8#;
      if PARITY = 'E' then
      LINE := LINE + 16#10#;
```

```
      end if;                          -- default parity : odd
    end if;

    if PORT = 1 then
      INT_MASK := 16#EF#;              -- reset bit 4
    else
      INT_MASK := 16#F7#;              -- default port2 & reset bit 3
    end if;

    if PORT = 1 then
      INT_NUM := 16#0C#;               -- SERIAL 1 INTERRUPT #, 0CH (12)
    else
      INT_NUM := 16#0B#;               -- default SERIAL2 INT #, 0BH (11)
    end if;

    INIT_SERIAL(RX_REG,TX_REG,INT_EN,LINE_CRT,MODEM_CRT,LINE_STAT,
            BAUD_LSB,LINE,INT_MASK,INT_NUM);
  end OPEN_SERIAL;

end SERIAL;
```

```
;*****************************************************************
;
;   File Name : SERIAL.ASM
;   Author : Se-Hung Kwak
;   DATE   : 9/11/91
;
;*****************************************************************

      NAME   SERIAL
      DGROUP  GROUP  DATA

data    segment para public 'data'
bufsiz  equ     4096
buffer  db      bufsiz dup(0)           ; buffer
bufptr1 dw      0                       ; points to start of buffer
bufptr2 dw      0                       ; points to end of buffer
bufcs   dw      0                       ; interrupt vector cs buffer
bufip   dw      0                       ; interrupt vector ip buffer

RX_REG  DW      0                       ; RX REG ADDRESS
TX_REG  DW      0                       ; TX REG ADDRESS
INT_EN  DW      0                       ; INT ENABLE REG ADDRESS
LINE_CRT  DW    0                       ; LINE CRT REG ADDRESS
MODEM_CRT DW    0                       ; MODEM CRT REG ADDRESS
LINE_STAT DW    0                       ; LINE STAT REG ADDRESS
BAUD_LSB  DW    0                       ; BAUD DIVISOR (LSB)
LINE      DW    0                       ; LINE VALUE (DATA BIT,PARITY,STOP)
INT_MASK  DW    0                       ; 8259 INT MASK VALUE FOR PORT1 OR 2
INT_NUM   DW    0                       ; INTERRUPT NUMBER FOR PORT1 OR 2
data    ends
;

_SERIAL  segment para public 'code'
      ASSUME CS:_SERIAL, DS:DGROUP


;*********************************************************************
;
; Procedure INIT_SERIAL(RX_REG,TX_REG,INT_EN,LINE_CRT,MODEM_CRT,
;                       LINE_STAT,BAUD_LSB,LINE,INIT_MASK,
;                       INT_NUM : in INTEGER);
;
;*********************************************************************


      PUBLIC  INIT_SERIAL

INIT_SERIAL   PROC  FAR
;
      cli                       ;disable all interrupts
      PUSH       BP
      MOV        BP, SP
      PUSH       DS
      MOV        AX, DATA
      MOV        DS, AX
      PUSH       DX
      PUSH       DI
      PUSH       SI
```

```
        PUSH        CX
      mov         ax, [BP+6]        ; GET RX_REG ADDR
      mov         RX_REG, ax
      mov         ax, [BP+8]        ; GET TX_REG ADDR
      mov         TX_REG, ax
      mov         ax, [BP+10]       ; GET INT_EN ADDR
      mov         INT_EN, ax
      mov         ax, [BP+12]       ; GET LINE_CRT ADDR
      mov         LINE_CRT, ax
      mov         ax, [BP+14]       ; GET MODEM_CRT ADDR
      mov         MODEM_CRT, ax
      mov         ax, [BP+16]       ; GET LINE_STAT ADDR
      mov         LINE_STAT,ax
      mov         ax, [BP+18]       ; GET BAUD_LSB
      mov         BAUD_LSB,ax
      mov         ax, [BP+20]       ; GET LINE
      mov         LINE, ax
      mov         ax, [BP+22]       ; GET INT_MASK
      mov         INT_MASK, ax
      mov         ax, [BP+24]       ; GET INT_NUM
      mov         INT_NUM, ax
;
;  set baud
;
      mov         dx, LINE_CRT      ; select baud divisor
        mov         ax, dx
      mov         al, 80h
      out         dx, al
      mov         dx, RX_REG        ; LSB divisor
      mov         ax, BAUD_LSB
      out         dx, al
      mov         dx, INT_EN        ; MSB divisor
      mov         al, 0
      out         dx, al
;
; init line control reg.
;
      mov         dx, LINE_CRT
      mov         ax, LINE
      out         dx, al
;
; init modem control reg.
;
      mov         dx, MODEM_CRT
      mov         al, 0Bh           ; loop back test
      out         dx, al
;
; enable interrupts
;
      mov         dx, INT_EN
      mov         al, 1           ; enabled receiver-data-ready
      out         dx, al
;
;   save interrupt vector
;
        push        es              ; es:bx vector will be returned
        push        bx
      mov         ax, INT_NUM ; give interrupt number
```

```
      mov       ah, 35h     ; dos function call #35h: get vector
      int       21h         ; dos function call int 21h
        mov       bufip, bx   ; save ip
        mov       bufcs, es   ; save cs
        pop       bx
      pop       es
      mov       cx, INT_NUM ; save INT_NUM into CX because of DS change
;
;   Set up interrupt vector table
;
        push      ds          ; ds:dx will be saved into vector table
        push      dx
        mov       ax, offset asyint
        mov       dx, ax
        mov       ax, cs
        mov       ds, ax
      mov       ax, CX      ; give interrupt number
      mov       ah, 25h     ; dos function call #25h: Set int vector
        int       21h         ; dos function call int 21h
        pop       dx
      pop       ds
;
;  adjust interrupt mask reg in 8259
;
      in        al, 21h      ; interrupt mask pattern
      and       ax, INT_MASK ; enable irq3 or 4 by resetting proper bit
      out       21h, al      ; save to interrupt mask reg in 8259

        POP       CX
      POP       SI
      POP       DI
      POP       DX
      POP       DS
        POP       BP
      sti
      RET       20
INIT_SERIAL   ENDP


;*************************************************************************
;
; Procedure CLOSE_SERIAL
;
;*************************************************************************


      PUBLIC  CLOSE_SERIAL

CLOSE_SERIAL   PROC  FAR
      cli                   ;disable all interrupts
      PUSH      DS
      MOV       AX, DATA
      MOV       DS, AX
      PUSH      CX
;
;  adjust interrupt mask reg in 8259
;
```

```
        push    bx
        mov     bx, INT_MASK ; get INT_MASK pattern
        not     bx           ; flip INT_MASK pattern
        mov     ax, bx
        in      al, 21h      ; interrupt mask pattern
        or      ax, bx       ; disable irq3 or 4 by setting proper bit
        out     21h, al      ; save to interrupt mask reg in 8259
        pop     bx

        MOV     CX, INT_NUM  ; save INT_NUM into CX because of DS change

;
; restore  interrupt vector for serial-2
;
          push    ds          ; ds:dx will be saved into vector table
          push    dx
          mov     dx, bufip
        mov     ds, bufcs
        mov     ax, CX      ; get proper INTERRUPT NUMBER
        mov     ah, 25h     ; dos function call #25h: Set int vector
          int     21h         ; dos function call int 21h
          pop     ds
        pop     dx

        POP     CX
        POP     DS
        sti                 ; enable all interrupts
        RET
CLOSE_SERIAL   ENDP


;*************************************************************************
;
; Procedure READ_CHAR(CH, DATA_READY : out CHARACTER);
;                     DATA_READY = 'Y' New CH
;                     DATA_READY = 'N' NO  CH
;*************************************************************************

        PUBLIC  READ_CHAR
READ_CHAR   PROC   FAR
        STI
        PUSH    BP
        MOV     BP, SP
        PUSH    DS
        MOV     AX, DATA
        MOV     DS, AX
        call    chget
        mov     bx, ax              ; save received char
        mov     al, ah
        PUSH    ES
        LES     SI, DWORD PTR [BP+10]
        MOV     AL, 'N'
        MOV     ES:[SI], AL         ; DATA_READY = N
        CMP     AH, 0
        JE      R_END               ; NO Ch Available -> return
        LES     SI, DWORD PTR [BP+10]
        MOV     AL, 'Y'
        MOV     ES:[SI], AL         ; YES, ch. DATA_READY = Y
```

```
      LES         SI, DWORD PTR [BP+6]
      mov         ax, bx                ; restore received char
      MOV         ES:[SI], AL           ; Return CH
R_END:  POP         ES
      POP         DS
      POP         BP
      RET         8
READ_CHAR   ENDP


;*************************************************************************
;
; Function WRITE_CHAR(CH: in CHARACTER) return CHARACTER
;                     Return 'Y'  CH is out
;                     Return 'N'  CH is not out. Buffer is full
;*************************************************************************

      PUBLIC  WRITE_CHAR
WRITE_CHAR   PROC   FAR
      STI
      PUSH        BP
      MOV         BP, SP
      PUSH        DS            ; Save DS
      MOV         AX, DATA      ; Data is accessable
      MOV         DS, AX
      MOV         CL, 'N'       ; TX buf is full
      MOV         DX, LINE_STAT   ; Line Status Reg
      IN          AL, DX
      TEST        AL, 20H       ; TX is empty?
      JZ          W_END         ; Not empty, return
      MOV         AL, [BP+6]    ; Get CHAR
      MOV         DX, TX_REG    ; Output to TX Reg
      OUT         DX, AL
      MOV         CL, 'Y'       ; Success
W_END:
      POP         DS
      POP         BP
      RET         2

WRITE_CHAR  ENDP

;
; serial communication interrupt routine
;
asyint  proc    far
      push    dx
      push    bx
      push    ax

;
; place the ascii char into the buffer.
;
      cli
        push    ds
        mov     ax, data
        mov     ds, ax
      mov     dx, RX_REG              ; read data port
      in      al,dx
```

```
      and     al,7fh                  ; strip off bit 7
      mov     bx,bufptr2              ; bx <- bufptr2
      mov     [buffer+bx], al         ; save into buffer
      inc     bx                      ; inc ptr2
      cmp     bx,bufsiz               ; end of buffer ?
      jc      asyskip                 ; no
      mov     bx,0                    ; yes, wrap around
asyskip: cmp    bx,bufptr1              ; buffer full ?
      jz      end_asy                 ; yes, ignore input data
      mov     bufptr2,bx              ; save ptr2 into bufptr2
;
end_asy: mov    al,20h                  ; send EOI (end of interrupt) command
      out     20h,al                  ; to port 20 (8259 command reg)
        pop     ds
      sti
      pop     ax
      pop     bx
      pop     dx
      iret
asyint  endp
;
;
;
; get character (al <- data,  ah <- 1 : success, ah <- 0 : buffer empty)
;
chget   proc       near
      push       bx
;        cli                             ; disable all interrputs
      mov        bx,bufptr1              ; get ptr1
      cmp        bx,bufptr2              ; buffer empty ?
      jnz        chget2                  ;
      mov        ah,0                    ; no char in the buffer
      jmp        chgete                  ; get out from chget
;
chget2: mov        al,[buffer+bx]          ; NO, pass char through al reg
      inc        bx                      ; inc ptr1
      cmp        bx, bufsiz              ; end of buffer ?
      jc         chget3                  ;
      mov        bx,0                    ; YES, reset ptr1
chget3: mov        bufptr1,bx              ; save ptr1
      mov        ah,1                    ; success
chgete:
      ;sti                               ; enable int
      pop        bx
      ret
chget   endp

;
;
;
; display a char on the screen in the al reg with ascii format
;
;
disply proc       near
      push       bx
      push       ax           ; save char

;
```

```
; prepare to display the char.
;
      mov         bx,0
      mov         ah,14
      int         10h
      pop         ax
      push        ax
      cmp         al,0dh
      jnz         end_dis
;
; return -> return + line feed
;
      mov         al,0ah
      mov         bx,0
      mov         ah,14
      int         10h
end_dis: pop        ax
      pop         bx
      ret
disply endp
;

_SERIAL   ends
      end
```

```
--*****************************************************************
--
--    File Name : TOKEN_DE.A
--    Author : Se-Hung Kwak
--    DATE   : 9/11/91
--
--*****************************************************************

with TEXT_IO;
use  TEXT_IO;

package TOKEN is


  procedure GET_TOKENT (FILE : in out FILE_TYPE;
                        STARTC, ENDC, SPACER: in CHARACTER;
                        TOKEN: out STRING; TOKEN_SIZE: out INTEGER;
                        FRAME_END: out BOOLEAN;
                        END_OF_STREAM: out BOOLEAN);
  procedure GET_FRAMET(FILE : in out FILE_TYPE;
                       STARTC, ENDC : in CHARACTER;
                       FRAME : out STRING; FRAME_SIZE : out INTEGER;
                       END_OF_STREAM: out BOOLEAN);
  procedure GET_FRAME(
                    STARTC, ENDC : in CHARACTER;
                    FRAME : out STRING; FRAME_SIZE : out INTEGER;
                    END_OF_STREAM: out BOOLEAN);
  procedure GET_TOKEN(FRAME: in STRING; FRAME_SIZE: in INTEGER;
                      STARTC, SPACER, ENDC: in CHARACTER;
                      POS_PTR : in out INTEGER;
                   TOKEN: out STRING; TOKEN_SIZE: out INTEGER);


end TOKEN;
```

```
--*****************************************************************
--
--    File Name : TOKEN.A
--    Author : Se-Hung Kwak
--    DATE   : 9/11/91
--
--*****************************************************************

with SERIAL, STR;
use  SERIAL, STR;

package body TOKEN is

  START_FLAG : BOOLEAN := FALSE;


  procedure GET_TOKENT (FILE: in out FILE_TYPE;
                        STARTC, ENDC, SPACER: in CHARACTER;
                        TOKEN: out STRING; TOKEN_SIZE: out INTEGER;
                        FRAME_END: out BOOLEAN;
                        END_OF_STREAM: out BOOLEAN) is
    CH : CHARACTER;
    PTR : INTEGER;
  begin
    PTR := 0;
    END_OF_STREAM := FALSE;
    if START_FLAG = FALSE then
      GET(FILE,CH);
      while not (CH = STARTC) loop
        GET(FILE,CH);
      end loop;
      START_FLAG := TRUE;
    end if;
    GET(FILE,CH);
    while not ( CH = SPACER or CH = ENDC) loop
      PTR := PTR + 1;
      TOKEN(PTR) := CH;
      GET(FILE,CH);
    end loop;
    if CH = ENDC then
      START_FLAG := FALSE;
    end if;
    TOKEN_SIZE := PTR;
    FRAME_END := not START_FLAG;
    exception
      when END_ERROR =>
        close(FILE);
        END_OF_STREAM := TRUE;
  end GET_TOKENT;

  procedure GET_FRAMET(FILE : in out FILE_TYPE;
                       STARTC, ENDC : in CHARACTER;
                       FRAME : out STRING; FRAME_SIZE : out INTEGER;
                       END_OF_STREAM: out BOOLEAN) is
    CH : CHARACTER;
    PTR : INTEGER;
  begin
    PTR := 0;
```

```
  END_OF_STREAM := FALSE;
  GET(FILE,CH);
  while not (CH = STARTC) loop
    GET(FILE,CH);
  end loop;
  PTR := PTR + 1;
  FRAME(PTR) := CH;  -- save start char into frame
  GET(FILE,CH);
  while not (CH = ENDC) loop
    PTR := PTR + 1;
    FRAME(PTR) := CH;
    GET(FILE,CH);
  end loop;
  PTR := PTR + 1;
  FRAME(PTR) := CH; -- save end character into frame
  FRAME_SIZE := PTR;
  exception
    when END_ERROR =>
      close(FILE);
      END_OF_STREAM := TRUE;
end GET_FRAMET;

procedure GET_FRAME(
                  STARTC, ENDC : in CHARACTER;
                  FRAME : out STRING; FRAME_SIZE : out INTEGER;
                  END_OF_STREAM: out BOOLEAN) is
  CH, FLAG : CHARACTER;
  PTR : INTEGER;
begin
  PTR := 0;
  END_OF_STREAM := FALSE;
  loop
    READ_CHAR(CH,FLAG);
    exit when FLAG = 'Y';
  end loop;
  while not (CH = STARTC) loop
    loop
    READ_CHAR(CH,FLAG);
    exit when FLAG = 'Y';
    end loop;
  end loop;
  PTR := PTR + 1;
  FRAME(PTR) := CH;  -- save start char into frame
  loop
    READ_CHAR(CH,FLAG);
    exit when FLAG = 'Y';
  end loop;
  while not (CH = ENDC) loop
    PTR := PTR + 1;
    FRAME(PTR) := CH;
    loop
    READ_CHAR(CH,FLAG);
    exit when FLAG = 'Y';
    end loop;
  end loop;
  PTR := PTR + 1;
  FRAME(PTR) := CH; -- save end character into frame
  FRAME_SIZE := PTR;
```

```
    exception
      when END_ERROR =>
      END_OF_STREAM := TRUE;
  end GET_FRAME;

  procedure GET_TOKEN(FRAME: in STRING; FRAME_SIZE: in INTEGER;
                      STARTC, SPACER, ENDC: in CHARACTER;
                      POS_PTR : in out INTEGER;
                      TOKEN: out STRING; TOKEN_SIZE: out INTEGER) is
    PTR : INTEGER;
  begin
    PTR := 1;
    if FRAME(POS_PTR) = STARTC then
      POS_PTR := POS_PTR + 1;
    end if;       -- Skip start character
    TOKEN(PTR) := FRAME(POS_PTR);
    POS_PTR := POS_PTR + 1;
    while not ((FRAME(POS_PTR) = SPACER) or (FRAME(POS_PTR) = ENDC)) loop
      PTR := PTR + 1;
      TOKEN(PTR) := FRAME(POS_PTR);
      POS_PTR := POS_PTR + 1;
    end loop;
    TOKEN_SIZE := PTR;
  end GET_TOKEN;


end TOKEN;
```

```
--*****************************************************************
--
--    File Name : TRIMBLE_.A
--    Author : Se-Hung Kwak
--    DATE   : 9/11/91
--
--*****************************************************************

with TEXT_IO, GPS;
use  TEXT_IO, GPS;

package TRIMBLE is
  procedure GET_TRIMBLE_DATA(
                              GPS_DATA: out GPS_DATA_TYPE;
                              GPS_RAW_DATA : in out STRING;
                              RAW_DATA_SIZE : in out INTEGER;
                              END_OF_DATA : out BOOLEAN);

  procedure GET_TRIMBLE_DATAT(INF: in out FILE_TYPE;
                         GPS_DATA: out GPS_DATA_TYPE;
                         GPS_RAW_DATA : in out STRING;
                         RAW_DATA_SIZE : in out INTEGER;
                         END_OF_DATA : out BOOLEAN);
end TRIMBLE;
```

```
--****************************************************************
--
--    File Name : TRIMBLE.A
--    Author : Se-Hung Kwak
--    DATE   : 9/11/91
--
--****************************************************************

with TOKEN, STR;
use  TOKEN, STR;
with scr; use scr;

package body TRIMBLE is

  type MONTH_OF_THE_YEAR is (JAN, FEB, MAR, APR, MAY, JUN,
                        JUL, AUG, SEP, OCT, NOV, DEC);
  type DAY_OF_THE_WEEK is (SUN, MON, TUE, WED, THU, FRI, SAT);

  -- Trimble's start end frame chars are '[' and ']'.
  -- Trimble's spacer is ' '.

  STARTC : constant CHARACTER := '[';
  ENDC   : constant CHARACTER := ']';
  SPACE  : constant CHARACTER := ' ';
  NUM_OF_TOKEN : constant INTEGER := 17;  -- number of items in one frame


  function DEG_MIN_TO_DEG(DIN : in STRING; SIZE: in INTEGER) return FLOAT is
  -- North and East is positive
    DEG_STR,MIN_STR : STRING(1..256);
    DEG, MIN : FLOAT;
    DIR : CHARACTER;
    POS : INTEGER;
  begin
    SPLIT_STR(DIN,SIZE,':',DEG_STR,MIN_STR,POS);
    DEG := FLOAT(INTEGER'VALUE(DEG_STR(1..POS-1)));
    DIR := MIN_STR(SIZE-POS);
    MIN := STR_TO_FLOAT(MIN_STR(1..SIZE-POS-1),SIZE-POS-1);
    if DIR = 'W' or DIR = 'S' then
      return -1.0 * (DEG + MIN/60.0);
    else
      return DEG + MIN/60.0;
    end if;
  end DEG_MIN_TO_DEG;

  procedure TIME_TO_H_M_S(DIN : in STRING; SIZE: in INTEGER;
                    HOUR, MIN, SEC : out INTEGER) is
    HOUR_STR, MIN_SEC_STR, MIN_STR, SEC_STR : STRING(1..10);
    MIN_SEC_SIZE, POS : INTEGER;
  begin
    SPLIT_STR(DIN,SIZE,':', HOUR_STR, MIN_SEC_STR,POS);
    HOUR := INTEGER'VALUE(HOUR_STR(1..POS-1));
    MIN_SEC_SIZE := SIZE-POS;
    SPLIT_STR(MIN_SEC_STR,MIN_SEC_SIZE,':',MIN_STR,SEC_STR,POS);
    MIN := INTEGER'VALUE(MIN_STR(1..POS-1));
    SEC := INTEGER'VALUE(SEC_STR(1..MIN_SEC_SIZE-POS));
  end;
```

```
procedure DATE_TO_Y_M_D(DIN: in STRING; SIZE: in INTEGER;
                        YEAR, MONTH_NUM, DAY : out INTEGER) is
  DAY_STR, MON_YEAR_STR, MON_STR, YEAR_STR : STRING(1..10);
  MONTH : MONTH_OF_THE_YEAR;
  MON_YEAR_SIZE, POS : INTEGER;
begin
  SPLIT_STR(DIN,SIZE,'-',DAY_STR, MON_YEAR_STR, POS);
  DAY := INTEGER'VALUE(DAY_STR(1..POS-1));
  MON_YEAR_SIZE := SIZE-POS;
  SPLIT_STR(MON_YEAR_STR,MON_YEAR_SIZE,'-',MON_STR,YEAR_STR,POS);
  MONTH := MONTH_OF_THE_YEAR'VALUE(MON_STR(1..POS-1));
  MONTH_NUM := MONTH_OF_THE_YEAR'POS(MONTH) + 1;  -- FIRST POS VALUE is 0.
  YEAR := INTEGER'VALUE(YEAR_STR(1..MON_YEAR_SIZE - POS));
end;
```

```
procedure GET_TRIMBLE_DATA(
                      GPS_DATA: out GPS_DATA_TYPE;
                       GPS_RAW_DATA : in out STRING;
                       RAW_DATA_SIZE : in out INTEGER;
                       END_OF_DATA : out BOOLEAN) is
  YEAR_NUM : INTEGER;
  TOKEN  : STRING(1..256);
  TOKEN_SIZE, POS_PTR: INTEGER;
  END_OF_STREAM : BOOLEAN;
  x : float;

begin
  GET_FRAME(STARTC,ENDC,GPS_RAW_DATA,RAW_DATA_SIZE,END_OF_STREAM);
  if END_OF_STREAM then
    END_OF_DATA := TRUE;
  else
    END_OF_DATA := FALSE;
    POS_PTR := 1;  -- character pointer inside a frame
    for COUNTER in 1..NUM_OF_TOKEN loop
    GET_TOKEN(GPS_RAW_DATA,RAW_DATA_SIZE,STARTC,SPACE,ENDC,
            POS_PTR,TOKEN,TOKEN_SIZE);
    if COUNTER = 4 then
      DATE_TO_Y_M_D(TOKEN(1..TOKEN_SIZE),TOKEN_SIZE,
                YEAR_NUM,
                GPS_DATA.DATE.MONTH,
                GPS_DATA.DATE.DAY);
      if YEAR_NUM > 50 then
        GPS_DATA.DATE.YEAR := 1900 + YEAR_NUM;
      else
         YEAR_NUM := 2000 + YEAR_NUM;
      end if;
    elsif COUNTER = 5 then
      TIME_TO_H_M_S(TOKEN(1..TOKEN_SIZE),TOKEN_SIZE,
                GPS_DATA.TIME.HOUR,
                GPS_DATA.TIME.MINUTE,
                GPS_DATA.TIME.SECOND);
    elsif COUNTER = 6 then
      GPS_DATA.POSITION.LATITUDE := DEG_MIN_TO_DEG(TOKEN(1..TOKEN_SIZE),
```

```
                                          TOKEN_SIZE);
    elsif COUNTER = 7 then
      GPS_DATA.POSITION.LONGITUDE := DEG_MIN_TO_DEG(TOKEN(1..TOKEN_SIZE),
                                          TOKEN_SIZE);
    elsif COUNTER = 8 then
      GPS_DATA.POSITION.ALTITUDE :=
            FLOAT(INTEGER'VALUE(TOKEN(1..TOKEN_SIZE)));
    elsif COUNTER = 11 then
      GPS_DATA.VELOCITY.X_VELOCITY := STR_TO_FLOAT(TOKEN(1..TOKEN_SIZE),
                                          TOKEN_SIZE);
    elsif COUNTER = 12 then
      GPS_DATA.VELOCITY.Y_VELOCITY := STR_TO_FLOAT(TOKEN(1..TOKEN_SIZE),
                                          TOKEN_SIZE);
    elsif COUNTER = 9 then
      GPS_DATA.PDOP := STR_TO_FLOAT(TOKEN(1..TOKEN_SIZE),
                           TOKEN_SIZE);
    else
      null;
    end if;
    end loop;
  end if;
end GET_TRIMBLE_DATA;


procedure GET_TRIMBLE_DATAT(INF : in out FILE_TYPE;
                     GPS_DATA: out GPS_DATA_TYPE;
                      GPS_RAW_DATA : in out STRING;
                            RAW_DATA_SIZE : in out INTEGER;
                            END_OF_DATA : out BOOLEAN) is
  YEAR_NUM : INTEGER;
  TOKEN  : STRING(1..256);
  TOKEN_SIZE, POS_PTR: INTEGER;
  END_OF_STREAM : BOOLEAN;

begin
  GET_FRAMET(INF,STARTC,ENDC,GPS_RAW_DATA,RAW_DATA_SIZE,END_OF_STREAM);
  if END_OF_STREAM then
    END_OF_DATA := TRUE;
  else
    END_OF_DATA := FALSE;
    POS_PTR := 1;  -- character pointer inside a frame
    for COUNTER in 1..NUM_OF_TOKEN loop
      GET_TOKEN(GPS_RAW_DATA,RAW_DATA_SIZE,STARTC,SPACE,ENDC,
                POS_PTR,TOKEN,TOKEN_SIZE);
      if COUNTER = 4 then
      DATE_TO_Y_M_D(TOKEN(1..TOKEN_SIZE),TOKEN_SIZE,
                YEAR_NUM,
                GPS_DATA.DATE.MONTH,
                GPS_DATA.DATE.DAY);
      if YEAR_NUM > 50 then
        GPS_DATA.DATE.YEAR := 1900 + YEAR_NUM;
      else
         YEAR_NUM := 2000 + YEAR_NUM;
      end if;
    elsif COUNTER = 5 then
        TIME_TO_H_M_S(TOKEN(1..TOKEN_SIZE),TOKEN_SIZE,
                      GPS_DATA.TIME.HOUR,
                      GPS_DATA.TIME.MINUTE,
```

```
                    GPS_DATA.TIME.SECOND);
        elsif COUNTER = 6 then
          GPS_DATA.POSITION.LATITUDE := DEG_MIN_TO_DEG(TOKEN(1..TOKEN_SIZE),
                                                       TOKEN_SIZE);
        elsif COUNTER = 7 then
          GPS_DATA.POSITION.LONGITUDE := DEG_MIN_TO_DEG(TOKEN(1..TOKEN_SIZE),
                                                        TOKEN_SIZE);
        elsif COUNTER = 8 then
          GPS_DATA.POSITION.ALTITUDE :=
                  FLOAT(INTEGER'VALUE(TOKEN(1..TOKEN_SIZE)));
        elsif COUNTER = 11 then
          GPS_DATA.VELOCITY.X_VELOCITY := STR_TO_FLOAT(TOKEN(1..TOKEN_SIZE),
                                                       TOKEN_SIZE);
        elsif COUNTER = 12 then
          GPS_DATA.VELOCITY.Y_VELOCITY := STR_TO_FLOAT(TOKEN(1..TOKEN_SIZE),
                                                       TOKEN_SIZE);
        elsif COUNTER = 9 then
          GPS_DATA.PDOP := STR_TO_FLOAT(TOKEN(1..TOKEN_SIZE),
                                        TOKEN_SIZE);
        else
          null;
        end if;
      end loop;
    end if;
  end GET_TRIMBLE_DATAT;

end TRIMBLE;
```

INITIAL DISTRIBUTION LIST

Defense Technical Information Center
Cameron Station
Alexandria, VA 22314 — 2

Dudley Knox Library
Code 52
Naval Postgraduate School
Monterey, CA 93943-5100 — 2

Office of Research Administration
Code 08
Naval Postgraduate School
Monterey, CA 93943-5100 — 1

Chief of Naval Research
800 N. Quincy Street
Arlington, VA 22302-0268 — 1

Center for Naval Analyses
4401 Ford Avenue
Alexandria, VA 22302-0268

Chief of Naval Operations — 1
Director, Information Systems (OP-945)
Navy Department
Washington, D.C. 20350-2000

Chairman, Code CS — 2
Computer Science Department
Naval Postgraduate School
Monterey, CA 93943-5100

Dr. Randy Brill — 2
NOSC Hawaii Laboratory
P.O. Box 997
Kailua, HI 96734-0997

Prof. R.B. McGhee, Code CS/Mz — 3
Deapartment of Computer Science
Naval Postgraduate School
Monterey, CA 93943

Professor Se-Hung Kwak, Code CS/Kw
Department of Computer Science
Naval Postgraduate School
Monterey, CA 93943